

FLAM fi

FRANKENSTEIN-LIMES-ACCESS-METHOD

(VSE fi)

BENUTZERHANDBUCH

— Ausgabe August 2008 Version 4.1 —

Telefon (06172) 5919-0 ■ Telefax (06172) 5919-39
http://www.flam.de ■ http://www.limes-datentechnik.de

Benutzerhandbuch FLAMfi V4.1 (VSE)

Vorwort

Dieses Handbuch beschreibt die Komprimierung und Dekomprimierung sowie die Verschl sselung von Daten mit der Frankenstein-Limes-Access-Method. Diese Methode wird durch das Produkt FLAM realisiert.

FLAM komprimiert strukturorientiert nach dem Algorithmus, der auch Bestandteil des in der Bundesrepublik Deutschland und in den USA sowie beim europ ischen Patentamt patentierten Frankenstein-Limes-Verfahrens ist; angemeldet durch die Erfinder am 19.07.1985.

FLAMfi, FLAMFILEfi und limes datentechnikfi sind eingetragene Warenzeichen/international trademarks.

Das Handbuch besteht aus folgenden Kapiteln:

Einf hrung
Hier werden Begriffe und Grundlagen erl utert und diverse Einsatzvorschl ge gemacht.

Funktionen
Alle Funktionen werden allgemein dargestellt.

Schnittstellen
Es werden alle Parameter und Programmschnittstellen formal beschrieben.

Arbeitsweise
Die interne Arbeitsweise wird soweit erkl rt, wie das f r den effizienten Einsatz des Produktes notwendig erscheint.

Anwendungsbeispiele
Es wird gezeigt, wie der Anwender z gig zu Ergebnissen kommen kann. Dazu gibt es diverse Beispiele, praktische Hinweise und Tips f r FLAM-Anwender.

Installation
Hier wird beschrieben, wie FLAM installiert wird und wie die Standardwerte ge ndert werden k nnen.

Technische Daten
Dieses Kapitel enth lt Angaben ber die Systemumgebung. Typische Leistungsmerkmale von FLAM werden dokumentiert.

Meldungen
Alle FLAM-Meldungen sind mit Bedeutung und Ma nahmen aufgef hrt.

Anhang
Hier sind die Code-Konvertierungstabellen sowie alle FLAM-Parameter aufgef hrt.

Welche Vorkenntnisse sind n tig?

Sie sollten ber VSE-Kenntnisse verf gen und insbesondere mit der Kommandosprache vertraut sein.

Als Unterlagen dienen Ihnen hierzu die Handb cher:

- SYSTEM CONTROL STATEMENTS
- VSE/VSAM COMMANDS

Die Neuerungen gegen ber dem Vorg ngermanual sind im ˜ nderungsprotokoll zusammengefasst.

Ein Literaturverzeichnis befindet sich im Anschluss an den Anhang.

limes datentechnikfi gmbh

FLAM (VSE)

Benutzerhandbuch

˜ nderungsprotokoll

˜ nderungsprotokoll - FLAM V4.1

˜ nderung des Manuals FLAM V3.0 durch diesen Nachtrag vom August 2008 (FLAM V4.1).

FLAM V4.1 ist eine Funktionserweiterung der Version 3.0. Sie ist aufw rtskompatibel zu allen Vorg ngerversionen. Die Komprimate der Versionen sind gleich und beliebig austauschbar, sofern keine neuen Funktionen oder Dateiformate dieser Version verwendet werden. Die AES-Verschl sselung z.B. setzt mindestens FLAM V4 voraus.

AES-Verschl sselung

Vom National Institute of Standards (NIST) wurde der **Advanced Encryption Standard** (AES**)** zur Verschl sselung von Daten festgelegt. Im November 2001 wurde dieses Verfahren im Federal Information Processing Standard (FIPS-197) beschrieben und mit Wirkung vom 26. Mai 2002 frei gegeben.

FLAM verwendet diesen Algorithmus zur Verschl s-selung der komprimierten Daten. Als Schl ssel k nnen bis zu 64 Zeichen angegeben werden. Intern wird eine Schl ssell nge von 128 Bit verwendet (AES-128). Zur Absicherung der Daten werden Kontrollfelder eingef gt (MACs), die ebenfalls mit AES gebildet werden.

Diese Verschl sselungsmethode wird aktiviert durch die Parameter CRYPTOMODE=AES und CRYPTO-KEY=key und ist nur f r die Kompressionsmethoden ADC und NDC (MODE=ADC bzw. MODE=NDC) implementiert.

Die Entschl sselung mit AES setzt FLAM V4.x voraus.

Sichern der FLAMFILE

Mit SECUREINFO=YES werden im ADC-/NDC-Modus zu-s tzliche Informationen im Komprimat gespeichert, die eine Vollst ndigkeit und Unversehrtheit des Komprimats sicherstellen, ohne dass die FLAMFILE dekomprimiert werden muss. Ist eine so gesicherte FLAMFILE ver ndert worden (z.B. durch Update, Erg nzen, L schen von Mem-bern in einer Sammel-FLAMFILE) wird dies bereits bei der formalen berpr fung erkannt.

Diese zus tzlichen Informationen werden bei Verschl s-selung immer geschrieben!

In FLAM V3 werden diese Informationen berlesen, sie f hren nicht zu einem Fehler in der Dekomprimierung.

Mit SECUREINFO=IGNORE k nnen bei der Dekomprimierung diese Daten ignoriert werden. Das ist z.B. sinnvoll bei konkatinierten gesicherten FLAMFILEs

oder wenn trotz einer Sicherheitsverletzung die Daten dekomprimiert werden sollen.
Mit SECUREINFO=MEMBER wird bei Dekomprimerung einzelner Member aus einer Sammel-FLAMFILE nur die Integrit t und Vollst ndigkeit dieser Member gepr ft.

Splitten der FLAMFILE

Es kann beim Komprimieren die zu erstellende FLAMFILE parallel oder seriell geteilt (gesplittet) werden. Die Steuerung erfolgt durch Parameter (SPLITMODE, SPLITNUMBER, SPLITSIZE).

Beim Dekomprimieren gen gt die Angabe des ersten Fragments der gesplitteten FLAMFILE. Parameter sind nicht notwendig. FLAM erkennt selbstt tig, ob und wie gesplittet wurde und sucht sich ggf. die zugeh rigen Fragmente selbst.

Splitten der FLAMFILE ist nur f r bin re Komprimate erlaubt (MODE=CX8,VR8,ADC,NDC), die einzelnen Fragmente der gesplitteten FLAMFILE enthalten zus tzliche bin re Informationen.

Serieller Splitt

Bei seriellem Splitt (SPLITMODE=SERIAL) wird nach Erreichen einer vorgegebenen Dateigr e (SPLITSIZE=*zahl in Megabyte*) die aktuelle Datei geschlossen und eine neue Datei erzeugt. Die Anzahl der so erzeugten Fragmente einer gesplitteten FLAMFILE ist nicht beschr nkt und h ngt nur von der erzeugten Datenmenge ab.

Beim Dekomprimieren pr ft FLAM die Reihenfolge, Vollst ndigkeit und Zusammengeh rigkeit der Fragmente.

Hiermit ist es z.B. m glich, Einschr nkungen bei Dateigr en (z.B. bei eMail-Attachments oder Filetransfer) zu unterst tzen. Auch k nnen so schon Fragmente im Netz bertragen werden, obwohl die Originaldatei noch nicht komplett komprimiert wurde.

Paralleler Splitt

Bei parallelem Splitt (SPLITMODE=PARALLEL) wird in eine vorgegebene Zahl von Dateien (SPLITNUMBER=*zahl*) komprimiert. In der vorliegenden Version k nnen bis zu 4 Dateien erzeugt werden. Die Dateigr e ist von der Menge der erzeugten Komprimatsdaten abh ngig.

Beim Dekomprimieren pr ft FLAM die Reihenfolge, Vollst ndigkeit und Zusammengeh rigkeit der Fragmente.

Eine Dekomprimierung ist nur m glich, wenn <u>alle</u> Fragmente der gesplitteten FLAMFILE im Zugriff sind. Fehlt auch nur ein Fragment, ist eine Dekomprimierung nicht m glich.

Mit dem parallelen Splitt ist es z.B. m glich, mehrere bertragungswege gleichzeitig zu bedienen und damit einen h heren Durchsatz zu erzielen. Aber auch eine sichere Datenarchivierung ist m glich. Verteilen der Fragmente auf verschiedene Orte verhindert eine Dekomprimierung, ohne dass eine Verschl sselung beim Komprimieren benutzt werden muss.

Pr fen der FLAMFILE

Mit dem Parameter CHECKFAST wird eine formale berpr fung der FLAMFILE vorgenommen. Dabei werden alle Checksummen, die Vollst ndigkeit und Integrit t der Daten berpr ft. Es erfolgt aber keine Dekomprimierung! Wird der Parameter CRYPTOKEY mit bergeben, werden zus tzlich s mtliche MACs gepr ft.

Mit dem Parameter CHECKALL wird wie bei CHECKFAST die FLAMFILE berpr ft, zus tzlich werden alle Daten dekomprimiert, ohne diese in eine Datei auszugeben. Bei einer verschl sselten FLAMFILE wird auch der Schl ssel ben tigt.

MODE=NDC

Mit dieser Methode (NDC=NoDataCompression) werden eingelesene Daten unkomprimiert verarbeitet. Sie sind dann nur entsprechend der FLAM-Syntax verpackt, gesichert und ggf. verschl sselt.

Hiermit wird Rechenzeit gespart bei Daten, die nur unwesentlich komprimiert werden k nnen, z.B. wenn FLAMFILEs erneut zusammengefasst werden sollen oder eine FLAMFILE erneut komprimiert und verschl sselt werden soll.

NDC ist kompatibel zu FLAM Version 3.

KMEXIT

Durch Einf hrung des Parameters KMEXIT wird der Anschluss des FLAM-Dienstprogramms an eine Schl sselverwaltung erm glicht. Damit wird eine Benutzerroutine aufgerufen, die zur Ver-/Entschl sselung einen Schl ssel zur Verf gung stellt.

KMPARM

Die KMEXIT-Routine kann ber den Parameter KMPARM Steueranweisungen f r den Ablauf erhalten. Zus tzliche Informationen k nnen bei der Verschl sselung in die FLAMFILE bernommen werden, die dem Exit bei der Entschl sselung wieder zur Verf gung stehen.

COMMENT

Mit dem Parameter COMMENT kann bei der Komprimierung mit dem Dienstprogramm ein Kommentar

in die FLAMFILE eingef gt werden. Dieser wird bei der Dekomprimierung im Protokoll ausgewiesen.

Erweiterung der Satzschnittstelle

Die Satzschnittstelle wurde um weitere Aufrufe erg nzt:

FLMEME

Beenden der eines Members in einer Sammel-FLAMFILE (End Member). Ggf. werden Sicherheitsinformationen in der FLAMFILE gespeichert (Membertrailer), bei AES-Verschl sselung wird der Hash-Mac dieses Members (Member-Mac) eingetragen und dem Aufrufer zur ckgegeben.

FLMSET

Erweitert die M glichkeit der Parameter bergabe. So k nnen hier Anweisungen zum Splitten der FLAMFILE oder der Verschl sselungsmethode bergeben werden.

FLMQRY

Erweitert die M glichkeit der Parameterabfrage. So k nnen hier Informationen zum Splitten der FLAMFILE oder zur Verschl sselung abgefragt werden.

Dateinamen

Um Konflikte mit unterschiedlichen Namenskonventionen anderer Betriebssysteme zu entsch rfen, werden die im ASCII-Code gespeicherten Dateinamen in einer FLAM-FILE zur Anzeige und Auswahl umgesetzt. Alle nationalen Sonderzeichen (wie ˜ {[]}) werden als gro es X dargestellt, der Backslash ‚\' umgesetzt in ‚/'. Leerzeichen werden zum Unterstrich ‚_'.
Entsprechend kann ein so umgesetzter Name (wie alle Parameter in Gro buchstaben) zur Dateiauswahl eingegeben werden.

Die Angabe *DUMMY als Parametereingabe f r Dateinamen ist implementiert. D.h. als Eingabedatei wird sofort auf EOF (End-of-File) verzweigt, als Ausgabedatei erfolgt keine Datenausgabe.

Meldungen

Die Protokollierung wurde um neue Meldungen erg nzt. Diese dienen der Information zum Integrit tsschutz, zum KMEXIT, zum Kommentar in der FLAMFILE und zum Splitting.

˜ nderungsprotokoll - FLAM V3.0A

˜ nderung des Manuals FLAM V2.5 durch diesen Nachtrag vom April 2000 (FLAM V3.0A).

FLAM V3.0A ist eine Funktionserweiterung der Version 2.5. Sie ist aufw rtskompatibel zu allen Vorg ngerversionen. Die Komprimate der Versionen sind gleich und beliebig austauschbar, sofern keine neuen Funktionen oder Dateiformate dieser Version verwendet werden.

Unterst tzung weiterer Dateiformate:

VSAM SAM-ESDS

SAM-ESDS-Dateien k nnen als Eingabe- oder als Ausgabedateien zugewiesen werden.

FLAM liest die Datei-Attribute aus dem VSAM-Catalog.

MEMBER aus LIBR-Bibliotheken

Es k nnen Member aus LIBR-Bibliotheken gelesen und geschrieben werden. FLAM unterst tzt beim Schreiben das RECORD-FORMAT mit FIX 80-BYTES und das BYTESTRING-FORMAT mit Recform=VAR oder UNDEF mit Satzl ngen bis 32760. Beim lesen im BYTESTRING-FORMAT muss der Anwender eine Satzl nge vorgeben andernfalls wird in Segmenten von 1024-Bytes gelesen. Member mit Type OBJ oder PROC k nnen nur FIX 80-Bytes gelesen und geschrieben werden. PHASEN werden von FLAM nicht unterst tzt (siehe 3.1.4).

MEMBER ausPOWER QUEUE

Es k nnen Member aus der READER / PUNCH / LIST-Queue gelesen und schrieben werden (siehe 3.1.4).

Automatisches Anlegen von VSAM-Dateien auch bei fehlender JCL

VSAM Dateinen k nnen automatisch, aber nur mit einem DATEN-Cluster bzw. bei KSDS mit einem DATEN-Cluster und einem INDEX-Cluster angelegt werden.
 ber Parametereingabe (FLAMIN=dateiname, FLAM-FILE=dateiname, FLAMOUT=dateiname) werden von FLAM f r VSAM-Dateien LABEL (DLBL) Bl cke erzeugt, sofern keine JCL vorgegeben ist.
Zum automatischen anlegen von VSAM Dateinen muss f r den jeweiligen Dateityp (SAM-ESDS,ESDS,...usw.) ein default Model im Katalog vorhanden sein (siehe Beispiel DEFMODEL.Z). F r bestehende VSAM Dateien haben FLAM-Dateiparameter keinen Einfluss, alle Dateiattribute bleiben erhalten.
Bei Angabe von FLAMOUT=<*> werden bei der Dekomprimierung alle Werte (wie Dateiname, Dateityp, Satzformat, Satz-, Blockl nge, Dateigr e (bei Komprimaten von VSE)) dem Fileheader der FLAMFILE entnommen.

Komprimieren vieler Dateien in eine FLAMFILE

Werden mehrere Dateien in eine FLAMFILE komprimiert, so sprechen wir von einer 'Sammeldatei'.

Ab dem jetzigen Release k nnen mehrere Dateien gleichzeitig in einem Aufruf komprimiert werden.

WILDCARD-Syntax

Durch Eingabe eines teilqualifizierten Dateinamens (z.B. VSEPSUC:USER.*.LST, FLAM.LIB(FL*.OBJ), ...) oder Vorgabe einer Dateiliste werden die ausgew hlten Dateien komprimiert in einer Komprimatsdatei (Sammeldatei) abgelegt (N:1-Beziehung).

Sammeldatei

Dabei werden alle VSAM-Dateien LIBR-MEMBER oder POWER-MEMBER durch FLAM dynamisch zugewiesen und der Dateityp (SAM-ESDS, VSAM-ESDS, VSAM-KSDS, VSAM-RRDS) sowie Satzformat (F, V, B, S, M, A) und Satz- und Blockl nge selbstt tig erkannt.

Aus dieser Sammeldatei k nnen die Dateien einzeln, selektiert nach Namen oder insgesamt dekomprimiert werden.

Beispiele:

Es sollen alle Dateien aus dem VSAM-Katalog VSESPUC mit 1. Qualifier USER und 3. Qualifier LST in die FLAMFILE VSESPUC:USER.CMP komprimiert werden:

```
// EXEC FLAM,PARM='C,FLAMIN=VSESPUC:USER.*.LST,
FLAMFILE=VSESPUC:USER.CMP,END'
```

Hier wird ein DD-Name als Eingabe zugewiesen. Die Datei enth lt Dateinamen, die in diesem Aufruf zu komprimieren sind:

```
// DLBL DIRIN,'DATEI.LISTE',,VSAM,CAT=VSESPUC
// DLBL FLAMFIL,'FLAMFILE',,VSAM,CAT=VSESPUC
// EXEC FLAM,PARM='C,IDDN=>DIRIN,END'
```

Komprimieren von vielen Dateien in viele Komprimatsdateien

Umsetzregeln f r Dateinamen (FLAM-FILEs)

Durch Eingabe eines teilqualifizierten Dateinamens (z.B. VSESPUC:USER.*.LST, USER.LIB(FL*.OBJ), ...) oder Vorgabe einer Dateiliste werden alle Dateien komprimiert in viele Komprimatsdateien abgelegt (N:N-Beziehung).

Der Name der FLAMFILE wird dann gem einer anzugebenden Umsetzvorschrift gebildet (z.B. FLAMFILE= <VSESPUC:*.LST=VSESPUC:*.CMP>, d.h. alle Dateien mit Endung LST erhalten die Endung CMP).

Umsetzregeln f r Dateinamen (FLAMOUT)

Durch Eingabe einer Umsetzvorschrift f r den Dateinamen der Dekomprimierung k nnen jetzt alle Dateien automatisch durch FLAM angelegt werden.

Es ist dabei nicht wichtig, ob das Komprimat unter einem fremden Betriebssystem (MVS, DPPX, UNIX, OS/2, ...) erstellt wurde. Alle Dateien werden in ein dem VSE-System entsprechendem Dateiformat erstellt.

Voraussetzung ist nur die Existenz des Fileheaders in der FLAMFILE (Parameter HEADER=YES (Default-Einstellung)).

Interne Dateinamen

Wurde eine Sammeldatei mit HEADER=YES aber FILE-INFO=NO (d.h. ohne Speicherung des Dateinamens) erzeugt, so kann jede Datei ber den internen Namen FILE0001 (f r die erste Datei) bis FILE9999 (f r die 9999. Datei) in einer Umsetzvorschrift angesprochen werden.

Der gespeicherte Dateiname kann bei der Dekomprimierung generell mittels FILEINFO=NO ignoriert werden. F r Umsetzregeln stehen dann die internen Namen zur Verf gung.

Neben weiteren Funktionen der Satzschnittstelle wurde die Performance beim Zugriff auf komprimierte Dateien ber die Satzschnittstelle verbessert (z.B. Laden im I/O-Modus, Masseneinf gungen bei KSDS).

Die Parametereingabe wurde bzgl. der Stringeingabe verbessert.

Erweiterung der Satzschnittstelle

Die Satzschnittstelle wurde um zwei Aufrufe erg nzt:

FLMIKY

Einf gen eines Satzes mit Schl ssel (Key) (bei existierendem Key wird nicht in die Datei geschrieben)

FLMLCR

sequentielles Lesen r ckw rts im Locate Mode

Verbesserung der Parametereingabe

Alle Strings (Dateinamen, Modulnamen, Satztrenner, ...) k nnen jetzt mit C'...' (d.h. Zeichendarstellung) oder X'...' (Hexwerte) eingegeben werden. Mit C'...' gekennzeichnete Strings k nnen Leerzeichen enthalten (z.B. FLAMOUT= <C'datei name'>).

Damit wird die unterschiedliche Namensgebung der verschiedenen Systeme (wie z.B. bei OS/2 oder UNIX) ber cksichtigt.

Neuer Parameter PADCHAR

Mit dem Parameter PADCHAR kann das F llzeichen zum Auff llen von S tzen bei der Dekomprimierung definiert werden.

Die Eingabe PADCHAR=X'00' bewirkt dann, dass die S tze der Originaldatei bei Konvertierung in eine gr ere (fixe) Satzl nge mit bin ren Nullen anstelle von Leerzeichen (default) aufgef llt werden.

Neue Komprimierungs-methode ADC

Mit *MODE=ADC* (**A**dvanced **D**ata **C**ompression) wird *"straight forward"* komprimiert. Die relative Optimierung zwischen verschiedenen Such- und Darstellungstechniken erfolgt flie end (adaptives Modell). Die Zuordnung der Codierung ndert sich *permanent*.

Komprimiert werden *autarke Datensegmente* von bis zu 64 KB. Der Anwender kann auf diese Gr e nur ber die Satzanzahl (MAXRECORDS) Einfluss nehmen. Die maximal zul ssige Satzanzahl wurde auf *4.095* erweitert (bisher *255*). MAXBUFFER ist 64 KB statisch (*ADC*).

Dieses Verfahren ist unabh ngig von einer Satzstruktur und zeigt h here Komprimierungsergebnisse als die Vorg ngerverfahren.

Neue Komprimatssyntax

Mit MODE=ADC unterscheidet sich jedes Komprimat (FLAMFILE) voneinander, auch bei identischer Eingabe. Damit wird die Sicherheit gegen eventuelle Angriffe von au en (bzw. Lesen auf der Leitung) erh ht. Zus tzlich kann so ein "Neukomprimieren" zwischendurch erkannt werden.

Mit MODE=ADC wurde eine neue Checksummentechnik eingef hrt, um den neuen File-Transfer-Produkten mit geringerer bertragungssicherheit Rechnung zu tragen.

Durch verschl sselte bernahme eines hardwarespezifischen Kennzeichens ist die (anonyme aber bestimmte) Herkunft einer FLAMFILE ermittelbar (sozusagen ein Quellenstempel, ohne aber die Quelle selbst preiszugeben).

Passwort

Mit MODE=ADC k nnen jetzt Komprimate mit einem Passwort versehen werden. Dieses Passwort kann bis zu 64 Zeichen (512 Bit) umfassen, es kann sowohl als abdruckbare Zeichen oder als Hex-String eingegeben werden. Mit diesem Passwort wird die FLAMFILE verschl sselt.

Erweiterung der Satzschnittstelle

Die Satzschnittstelle wurde um einen Aufruf erg nzt:

FLMPWD

bergabe eines Passwortes zur Komprimierung bzw. Dekomprimierung f r MODE=ADC.

Parameter

PASSWORD

PASSWORD zur Ver- bzw. Entschl sselung des Komprimats

PASSW

M gliche Werte

1 - 64 Zeichen in der Form C'...', X'...' oder als String

Standard:	kein Passwort
G ltig f r:	Komprimierung, Dekomprimierung

Hinweis: Bitte beachten Sie, dass abdruckbare Zeichen nationaler Zeichens tze auch bei der Dekomprimierung identisch (d.h. bin r-kompatibel) eingegeben werden m ssen. F r heterogenen Austausch empfehlen wir die hexadezimale Eingabe X'...'.

FLAM (VSE)

Benutzerhandbuch

Inhaltsverzeichnis

Anhang

FLAM (VSE)

Benutzerhandbuch

Kapitel 1:

Einf hrung

1. Einf hrung

FLAM ist eine Software zur Komprimierung von Daten, wie sie f r Applikationen von Banken, im Handel, in der Industrie und in der   ffentlichen Verwaltung typisch sind (tabellarische Daten).

FLAM komprimiert die im Kreditwesen normierten Formate des Datentr geraustausches etwa im Verh ltnis 4:1. Bei St cklisten liegt der Komprimierungseffekt nicht selten bei 95%.

FLAM ist keineswegs speziell f r den Einsatz im Kreditwesen entwickelt worden, obwohl es sich gerade im elektronischen Zahlungsverkehr zum optionalen Komprimierungsstandard entwickelt hat. Anwender nutzen FLAM wegen seiner vielf ltigen Einsatzm glichkeiten und der nachpr fbar kurzen Amortisationszeit.

FLAM bringt mit jeder neuen Einsatzvariante weitere Benefits, ohne dass zus tzliche Kosten entstehen. Folgerichtig ist es im Interesse jedes Anwenders, dazu beizutragen, dass immer mehr Hersteller und DF - Partner diese Technik unterst tzen. Das ist der besondere betriebswirtschaftliche Vorteil dieses Standards.

FLAM benutzt den dem ’Frankenstein-Limes-Verfahren zur strukturorientierten Datenkomprimierung’ zugrundeliegenden Algorithmus. Das so benannte Verfahren wurde in Deutschland, Europa und den USA patentiert. Die Anmeldung durch die Erfinder erfolgte am 19. Juli 1985.

FLAM arbeitet ohne Voranalyse und ohne Tabellentechnik. Dadurch ist die Dekomprimierung jederzeit aus dem Programm FLAM und der Syntax des Komprimats (FLAMFILE) heraus aufw rtskompatibel sichergestellt (Langzeitarchivierung).

FLAM ben tigt von au en keine Informationen   ber die zu komprimierenden Daten. Die Komprimierungstechnik ist invariant zu Datei-, Satz- und Feldformaten; die Komprimierungseffekte sind selbstverst ndlich abh ngig von den Dateninhalten. Strukturverzerrungen f hren meist zu schlechteren Komprimierungen.

FLAM erf llt als einziges Produkt dieser Art folgende Prinzipien:

Durchg ngigkeit

Die FLAM-Komprimate k nnen ohne Zwischenkonvertierungen zur Speicherung auf Online-Datentr gern in Verbindung mit sequentiellen und index-sequentiellen Zugriffsm glichkeiten benutzt werden. Ebenso durchg ngig sind FLAM-Komprimate zur Archivierung und zum File-Transfer im heterogenen Verbund, d.h. zwischen Rechnern mit unterschiedlichen Betriebssystemen geeignet.

Portabilit t

Die Komprimatsformatierung kann so gesteuert werden, dass alle Anforderungen an eine optimale Speicherbelegung sowie die Portabilit t auf beliebigen Leitungen unter Einsatz beliebiger File-Transfer-Produkte erf llbar sind. Dies gilt f r 'Lochkartenformate' (80-stellig) ebenso wie f r FTAM-Formate. Die Komprimatss tze k nnen im fixen oder variablen Format erzeugt werden.

Konvertibilit t

FLAM kann sogar Komprimate im abdruckbaren Format erzeugen, die zwischen Komprimierung und Dekomprimierung 1:1 von EBCDIC nach ASCII und umgekehrt konvertiert werden d rfen. Eine solche Konvertierung kann aber auch bei der Komprimierung/Dekomprimierung 'en passant' erledigt werden.

Kompatibilit t

FLAM konvertiert auf Wunsch Datei- und Satzformate. Dadurch kann FLAM Probleme der Konvertierung und der Kompatibilit t zwischen heterogenen Systemen oder versionsabh ngigen Datenverwaltungen l sen helfen. Restriktionen bez glich Satzformat (fix), doppelte Schl ssel u.a. neutralisiert die Zugriffsmethode FLAM.

Systemunabh ngigkeit

Eine FLAMFILE kann auf allen Systemen, f r die FLAM lieferbar ist, als Datenbasis f r die Zugriffsmethode FLAM benutzt werden, und zwar unter den verschiedenen systemspezifischen Zugriffsmethoden des betreffenden Datenverwaltungssystems.

Kontinuit t

Eine FLAMFILE kann beim Dekomprimieren in ein vom Anwender gew nschtes Datei-/Satzformat konvertiert werden. Damit ist die Kontinuit t garantiert. Eine archivierte FLAMFILE kann immer wieder auf irgendeinem System bearbeitet, insbesondere dekomprimiert werden. Eine Abh ngigkeit vom Betriebssystem besteht dann nicht mehr. Es muss gew hrleistet sein, dass der Datentr ger hardwarem ig gelesen werden kann und die FLAMFILE nicht in ein systemabh ngiges Format eines herstellerorientierten Archivierungsproduktes gebracht wurde.

Datensicherheit

FLAM verschleiert die Daten und versiegelt die Komprimate mittels Checksummen, womit die Daten besser gesichert und gesch tzt sind. Die FLAMFILE hat intern Synchronisationspunkte, um hinter Defekten, zum Beispiel durch Materialm digkeit, wieder aufsetzen zu k nnen. Forderungen der DV-Revision und des Datenschutzes werden voll erf llt.

Schnittstellen

FLAM bietet eine F lle von Schnittstellen, und zwar angelehnt an die Schnittstellen eines realen Datenverwaltungssystems mit index-sequentiellem Zugriff. FLAM kann als Unterprogramm komplett unter fremder Steuerung laufen. Benutzerausg nge von FLAM dienen der Vor-/Nachbehandlung der unkomprimierten Daten und FLAMFILE-S tze (Komprimatseinheiten).

Betriebssysteme

FLAM ist lieferbar f r die verschiedensten Betriebssysteme, wie z.B.:

FSC	BS2000/OSD Sinix Reliant Unix
HP	HPUX (div. Prozessoren) Windows Open VMS (DEC) True64 UNIX (DEC) Non Stop OS (TANDEM) OSS (TANDEM)
IBM	MVS, MVS-Subsystem, z/OS VM, VSE, z/VSE OS/400 AIX
NCR	Unix
SCO	SCO-Open Server SCO-UnixWare
SUN	SOLARIS
PCs	Windows Linux

Andere auf Anfrage.

Standard

FLAM ist optionaler Komprimierungsstandard f r diverse Verfahren im deutschen Kreditwesen, wie BCS, EAF (LZB), DTA u.a.

Hersteller

limes datentechnik gmbh

Philipp-Reis-Passage 2

D-61381 Friedrichsdorf/Ts.

Telefon 06172/5919-0

Telefax 06172/5919-39

eMail: info@flam.de
eMail: info@limesdatentechnik.de

Internet: http://www.flam.de
http://www.limes-datentechnik.de

Kooperationen

FLAM wird ber Interfaces zur Zeit von folgenden SW-Produkten unterst tzt:

BCS	Bank Verlag Gmbh

cfs	OPG Online Programmierung GmbH
MultiCom	CoCoNet AG
NATURAL	Software AG
SFIRM	BIVG Hannover GmbH & Co.KG
SIRON	Ton Beller AG

Manche Kooperationspartner bieten Interfaces ihrer SW-Produkte zu FLAM kostenpflichtig an. F r den Zahlungsverkehr (BCS) werden f r PC-Anwender komplette L sungen mit beschr nkter Anwendungsbreite ber Kreditinstitute und deren Partner angeboten.

Der Hersteller von FLAM ist f r jede weitere Kooperation mit Software-Herstellern auf der Basis der FLAM-Standards offen. Das bringt f r alle Beteiligten den optimalen Nutzen.

Die Vorteile von FLAM in Stichworten:

Datenfern bertragung

- Kostensenkung durch Mengenreduktion
- schnellere bertragung durch "Virtualisierung"
- implizite Beschleunigung anderer bertragungen
- Wechsel auf kosteng nstigere Leitungen m glich mit g nstigeren fixen Anschluss-/Betriebskosten
- weniger Fehler durch langsamere bertragungen, berwindung technologischer Engp sse (im Ausland)
- Erh hung der potentiellen Sende-/Empfangsfrequenz
- Entlastung von Netzknoten, Ports, Puffern und dgl.
- effizienteres Reagieren bei Leitungsst rungen sowie bei bertragungs- und Bedienungsfehlern m glich
- FLAMFILE in Parkplatzposition platzsparend und sofort restartf hig (Sender) und archivierbar
- Kompatibilit t der FLAMFILE im heterogenen Verbund
- Portabilit t der FLAMFILE durch Formatierbarkeit
- Konvertibilit t der FLAMFILE bei druckbaren Daten
- vor-/nachgeschaltete Zeichenkonvertierung m glich
- Konvertierung von Satz-/Dateiformaten (Utility)
- Durchg ngigkeit der FLAMFILE zu anderen Anwendungen
- mehr Fern berwachung/-wartung wg. Mengenreduktion
- mehr Datenaustausch per DF wg. Mengenreduktion
- mehr Auslagerungen in Not-RZ wg. Mengenreduktion
- Automatisierbarkeit von Fernarchivierungen (DF)
- Automatisierbarkeit des R cktransfers (analog)

- bessere DV-Revision durch Automatisierbarkeit
- mehr Datensicherheit durch Checksummen-Technik
- Datenschutz durch FLAM-typische Verschleierung
- h here Effizienz in Verbindung mit Kryptographie

Datenspeicherung

- Reduktion von Speicherplatz auf allen Medien mit weniger (sekund rem) Platzbedarf (r umlich)
- weniger Multi-Volumes-Files (Disc, Tape, Floppy)
- weniger Grundbedarf an Strom, Klima-, Schutzeinrichtungen, weniger Kapitalbindung (berkapazit t)
- weniger Overhead im Archiv und mehr Kontinuit t
- schnelleres I/O, resp. Entlastung der I/O-Kan le
- ggf. weniger Controller, I/O-Ports, Puffer
- Beschleunigung von Batch-/Kopier-Prozessen und Backup-/Restart-Verfahren, dadurch Reserven/Optionen f r mehr RZ-Automatisation/Redundanz
- Verk rzung von Ablaufzyklen, Anwesenheitszeiten
- zus tzlicher Zugriffsschutz durch FLAM-Processing
- integrierter Manipulationsschutz durch FLAM-Syntax
- verfahrensspezifische Datenverschleierung, sogar mit wirksamen Schutz f r "virtuell" gel schte Daten
- innovativ f r (kombinierte) Zugriffstechniken mit heterogen austauschbaren sequentiellen/index-sequentiellen Formaten sowie in logisch geblockten Einheiten

1.1 FLAMfi mit MODE=ADC

Seit FLAM V3.0 gibt es 3 essentielle Verbesserungen:

- einen universellen MODE=ADC (Advanced Data Compression)
- eine neue trickreiche FLAM-Syntax (Frankenstein-Limes-Access-Method)
- eine u erst effiziente PASSWORD-Verschl sselung.

Zun chst enth lt FLAM die vollst ndige Vorg ngerversion als Untermenge, so dass man einerseits mit MODE=CX7, CX8 und VR8 wie bisher de-/komprimieren kann; andererseits ist es dadurch unproblematisch, die betreffenden Komprimate zu erzeugen, weil etwa der Partner noch nicht auf FLAM V3.x umgestiegen ist. Dies betrifft Schnittstellen und User-Exits.

Die vorgenannten Modi zur Komprimierung haben bei den f r kommerzielle Anwendungen typischen Daten auf Mainframe au ergew hnlich gute Ergebnisse erbracht. Jeder Anwender kann selbst entscheiden, ob er bei dieser Technik bleiben will, wenn der Komprimierungseffekt ohnehin schon bei 85% oder mehr liegt.

Durch die st rkere Einbeziehung von PC- und UNIX-Systemen in die kommerzielle Datenverarbeitung haben sich die Datenstrukturen stark ver ndert. Die auf strukturelle Redundanzen ausgerichtete FLAM-Komprimierungstechnik musste auf kontextuelle Betrachtungen erweitert werden.

FLAM ist und bleibt ein als Zugriffsmethode konzipiertes Verfahren zum effizienten Umgang mit komprimierten Daten. Schon aus dieser Philosophie heraus darf FLAM keine tempor ren Dateien anlegen oder benutzen. Eine Voranalyse zur Auswahl geeigneter Komprimierungstechniken und/oder ein mehrstufiges Verfahren stehen im krassen Gegensatz zu den Anforderungen an eine performante Direkt-Zugriffsmethode (f r autarke Segmente), die in ihrem Kern invariant ber fast alle Plattformen hinweg konzipiert ist (vom PC bis zum Mainframe).

Der Anwender soll die Chance haben, so fr h, wie es sinnvoll erscheint, zu komprimieren, und so sp t wie n tig zu dekomprimieren, im Einzelfall (Retrieval) m glichst nur punktuell. Die FLAMFILEfi soll plattform bergreifend durchg ngig zur Speicherung, Archivierung und f r den File Transfer inkl. Backup (Auslagerung) als "Standard f r alle F lle" nutzbar sein.

Mit MODE=ADC (Advanced Data Compression) wird "straight forward" komprimiert. Die relative Optimierung

zwischen verschiedenen Such- und Darstellungstechniken erfolgt flie end (adaptives Modell). Die Zuordnung der Codierung ndert sich permanent.

Komprimiert werden autarke Datensegmente von bis zu 64 KB. Der Anwender kann auf diese Gr e nur ber die Satzanzahl (MAXRECORDS) Einfluss nehmen. Die maximal zul ssige Satzanzahl wurde auf 4.095 erweitert (bisher 255). MAXBUFFER ist 64 KB statisch (ADC).

Unter einem Satz wird eine im betreffenden Data Management System definierte logische Einheit verstanden. Es gibt fixe und variable Satzformate. Auf manchen Systemen haben die S tze ein L ngenfeld, auf anderen einen Delimiter. Das ist wichtig, wenn man aus der Sicht einer Anwendung oder beim Datenaustausch auf den Satz als logisch invariante Basis des Zugriffs angewiesen ist.

Auf Systemen, die keinen Dateikatalog mit Informationen ber das, was als Satz zu interpretieren ist, haben, kann man ohne weiteres auch einfach 64 KB einlesen, ohne dass diese Vorgehensweise die Komprimierung mit MODE=ADC nachteilig beeinflusst.

Wird eine Datei mit Delimiter auf PC oder UNIX gelesen und werden die Delimiter nicht als solche interpretiert, dann hat man beim Austausch im heterogenen Umfeld eventuell nach der Dekomprimierung das Problem der Anpassung an das betreffende Umfeld zu l sen.

Mit FLAM kann man bei Kenntnis und Nutzung des Satzformats mit der entsprechenden Parametrisierung diese Probleme von vornherein ausklammern. Damit hat man eine neutrale, zukunftssichere Darstellung, die sich beim Dekomprimieren automatisch den ge nderten Bedingungen anpassen l sst (Formatkonvertierung).

Nur mit FLAM kann man das Komprimat, die FLAMFILEfi , individuell formatieren, weil diese "Zwischendatei" ggf. ganz anderen Erfordernissen etwa in Verbindung mit File Transfer gen gen muss als die Originaldatei (Portabilit t).

Sind abdruckbare Daten so codiert, dass eine eindeutige Umcodierung 1:1 von EBCDIC nach ASCII oder umgekehrt m glich ist, dann kann dies beim De-/Komprimieren angesto en werden. Die mitgelieferten Tabellen dazu sind unverbindlich, weil es eine un bersichtliche Menge an Varianten dazu gibt. Es ist einfach, die betr. Tabelle auf die eigenen Bed rfnisse anzupassen. Wir empfehlen, auf dem System umzucodieren, auf dem dekomprimiert wird, weil dort erfahrungsgem die gr ere Sicherheit der relevanten Einstellung der Tabelle besteht. Damit sind Konvertibilit t und Kompatibilit t 1:1 sichergestellt.

F r den Datenaustausch in einem abdruckbaren Format mit einem File Transfer, der "unterwegs" umcodiert, muss man die Vorg ngerversion mit MODE=CX7 benutzen. Die Erfahrung hat gezeigt, dass die Umcodierung durch ein

File Transfer Produkt viele Unw gbarkeiten hat. Wir k nnen davon nur abraten. Die sichere L sung besteht im Austausch bin rer Daten und der Umcodierung davor oder (besser) danach. In aufbereiteten Drucklisten besteht zudem das Problem der Steuerung ber das erste Byte in jedem Satz (Drucksteuerzeichen).

Muss in ASCII bertragen werden, so stellen viele File Transfer Produkte Automatismen bereit, mit denen bin re Daten tempor r in scheinbar abdruckbare Daten umcodiert und nach der bertragung in den urspr nglichen Zustand gebracht werden. Man k nnte sich hierzu selbst eine Routine 3:4 schreiben und im User-Exit von FLAM aktivieren (Portabilit t).

Nicht selten treten in Verbindung mit File Transfer von FLAM-bierten Daten Formatfehler auf, die FLAM als Checksummenfehler meldet. Damit haben alle Beteiligten die Sicherheit, dass die bertragung aus Anwendersicht fehlerfrei abgelaufen ist (noch ber das FT-Protokoll hinaus). - Es gibt PC-Produkte, die haben erst gar keine Checksumme ber das Komprimat, sondern gerade eine einzige Checksumme ber die komplette Originaldatei, wobei die Originaldatei bis zu 4 GB gro sein darf. (FLAM hat keine Beschr nkung bzgl. Typ/Gr e.)

Es ist schon kurios: Ohne FLAM werden solche Fehler oft berhaupt nicht bemerkt, so dass nicht selten der falsche Eindruck entsteht, ein Fehler w rde ohne Beteiligung von FLAM nicht auftreten. Gerade die Kombination von FTP mit FLAM zeigt diesbez glich erstaunliche Synergieeffekte, die wegen mangelnder Sicherheit und Stabilit t im FTP unverzichtbar sind.

Es gibt eine ganze Reihe von Problemen in Verbindung mit File Transfer, die man in der Tat nur durch Einsatz von FLAM l sen kann. Ist das im Ausnahmefall nicht so, dann liegt das an dem Problem an sich und nicht an FLAM. So gibt es etwa gro e Probleme bei der Umcodierung von Zeichens tzen, wenn Sonderzeichen weitgehend ausgesch pft werden und dennoch nicht auf Umlaute verzichtet wird.

Man kann nicht komprimieren, ohne sich einen Arbeitsspeicher f r Hilfsinformationen anzulegen. F r MODE=ADC ben tigt FLAM ohne die Bereiche f r das I/O etwa 160 KB. Diese Grundmenge kann man aus der Sicht der Algorithmik nicht unterschreiten, wenn gleichzeitig ein vertretbarer Verbrauch an CPU-Zeit nicht berschritten werden soll. Im Vergleich zu anderen Modellen ist das f r ein adaptives Modell relativ wenig Arbeitsspeicher.

Bei einem Vergleich der Komprimierungseffekte mit anderen Produkten (meist PC-Produkte) m sste man fairer weise die Originaldatei zuvor in Segmente (kleine Dateien) von jeweils 64 KB aufteilen und die Einzelergebnisse aufaddieren. Au erdem hat eine FLAMFILE aus Sicherheitsgr nden wie auch wegen der innovativen Zugriffstechniken eine "Verpackung", die das Komprimat um bis zu 2% aufbl ht.

Die Beibehaltung der Segmentierung hat u.a. den Vorteil, dass bei schweren Datenfehlern ggf. nur ein einziges Segment betroffen ist. Jedes der Segmente in einer FLAMFILE wird autark betrachtet (quasi wie bei einer Transaktion) und als solches abgesichert (verpackt). Darauf kann man sich synchronisieren; man kann "mittendrin" an einem beliebigen Segment aufsetzen.

Zeigt sich w hrend des Komprimierungsvorgangs nach ca. 16 KB des betr. Segments gar kein Komprimierungseffekt, wird bei MODE=ADC die Komprimierung f r dieses Segment abgebrochen und der Original-Input von max. 64 KB (Segment) wird 1:1 bernommen.

Setzt in einem einzelnen Segment der Effekt erst nach 16 KB ein, wird dies nicht mehr erkannt, weil die Abw gung von Aufwand und Nutzen zu dem Schluss kommt, dass die Wahrscheinlichkeit, dieses Segment noch komprimieren zu k nnen, gering ist.

Denn: Je schlechter der Komprimierungseffekt, desto h her ist (leider) der CPU-Aufwand (weit berproportional). Das liegt in der Natur der Sache.

Mit einem Schichtenmodell sind in FLAM die Voraussetzungen geschaffen, Multiprozessorsysteme zu bedienen: ein Prozess liest, bildet die Segmente und verteilt sie zwecks Komprimierung an andere Prozesse; ein weiterer Prozess sammelt die komprimierten Segmente ein, formatiert sie zur FLAMFILE und schreibt diese.

Zur Zeit besteht zwar noch kein akuter Bedarf f r diese Vorgehensweise, aber das Modell in FLAM ist darauf vorbereitet.

FLAM "verweigert" sich nicht, wenn der Input selbst eine FLAMFILE ist. Das kann sogar eine sinnvolle Vorgehensweise sein. Man hat z.B. eine Bibliothek vieler kleiner Elemente, die zun chst autark komprimiert und als Sammeldatei abgelegt werden sollen, damit die Bibliothek mit ihren Elementnamen und deren Attributen ordnungsgem   rekonstruiert werden kann. In diesem Fall kann man nicht viel Komprimierung erwarten.

Nimmt man hierf r FLAM V2.x mit MODE=CX8 und MAXRECORDS=1, dann erf llt dieser Vorlauf nur den Zweck, die besagte Sammeldatei zu erstellen, bei der es mehr auf die diversen Informationen als auf den Komprimierungseffekt ankommt. Diese "flache" Datei l sst man durch FLAM V3.0 mit MODE=ADC komprimieren. - Anstelle des Vorlaufs mit FLAM V2.x kann man ggf. auch ein Utility benutzen, das eine ad quate Funktion erf llt (Sammeldatei).

In Ausnahmef llen gibt es sogar extrem stark strukturierte Dateien, die man zuvor mit FLAM V2.x, MODE=CX8 und MAXRECORDS=255 schon sehr gut komprimieren kann,

deren Komprimat sich dann mit FLAM V3.0 und MODE=ADC noch verbessern l sst. In der Regel aber ist FLAM V3.0 mit MODE=ADC und MAXRECORDS=4095 immer besser als die Vorg ngerversion oder eine zweistufige Variante damit. Es besteht kein Zwang, den Modus zu wechseln, wenn man mit der bisherigen Komprimierungstechnik und der Syntax in FLAM V2.x zufrieden ist. Neue Feature (z.B. die PASSWORD-Verschl sselung) setzen allerdings ausschlie lich auf FLAM V3.0 mit MODE=ADC auf, zumal die Syntax der FLAMFILE erheblich verbessert wurde.

Die neue Syntax garantiert einerseits, dass die Expansion bei Daten, die sich trotz ADC-Technik nicht komprimieren lassen, auf 2% beschr nkt bleibt; andererseits sind die in einem solchen Fall nur kopierten Originale nicht wiederzuerkennen.

Diese Eigenschaft hat ihre Ursache in einer weltweit einmaligen Checksummentechnik. Die vorletzte von 4 Checksummen (!) verschleiert parallel zur Checksummenbildung den komprimierten Input so, dass der Vorgang reversibel ist, wenn man die Checksummenfunktion zweimal anwendet. Sind die komprimierten Daten eines Segments verf lscht worden (Datenfehler, Manipulation), verbreitet sich der Defekt "wie die Pest" ber den Rest des komprimierten Segments. Die defekten Daten sind mithin danach unbrauchbar. Die Dekomprimierung l uft erst gar nicht an! Man kann diese CRC-Routine in FLAM auch nur starten, wenn das komprimierte Segment vollst ndig zur "Entschleierung" vorliegt.

Es gibt PC-Produkte, da kann man das Original "lesen", wenn nicht komprimiert wurde. CRC-Fehler werden erst nach dem CLOSE der dekomprimierten Datei gemeldet, weil die Checksumme auf den Originaldaten basiert. Die Dekomprimierung bricht trotz Checksummenfehler nicht vorzeitig ab. Die dekomprimierte Datei kann Fehler aller Art enthalten, sogar Abweichungen in der Gr e, obwohl im Header des Komprimats die richtige Byteanzahl steht.

In FLAM V3.0 mit MODE=ADC werden die Checksummen der Segmente ber einen Connector miteinander verkn pft. Wird nur seriell komprimiert und analog dekomprimiert, kann man die Unversehrtheit dieser Sequenz berpr fen.

Der Connector wird zudem mit einem zeitabh ngigen Code eingef rbt, so dass das gleiche Segment zu einem anderen Zeitpunkt komprimiert ein anderes "Outfit" bekommt. Der Komprimierungseffekt ndert sich nicht.

Eine weitere Modifikation besteht in einer sog. Hardware-ID. FLAM bildet aus Hardware-Informationen des Umfelds einen 32-Bit-Code. Dieser wird in den Connector eingearbeitet. Komprimiert man nun ein und dieselbe Datei zuf llig zu einem Zeitpunkt, der nicht zu einem Unterschied bei der Einstellung des Connectors f hrt, benutzt aber ein anderes Hardware-Umfeld, dann ndert

man dadurch zwangsl ufig den Connector und mithin wiederum das u ere Erscheinungsbild des Komprimats.

Ziel dieser Techniken ist es, dass m glichst jedes mit FLAM komprimierte Datensegment bez glich Inhalt (Original) sowie Umfeld und Zeitpunkt der Komprimierung eine Art Unikat sein soll. Die Checksummen der verschiedenen Schichten bilden in Summe eine Signatur, mit der ein Empf nger den Empfang zweifelsfrei quittieren k nnte (vollst ndig und unversehrt).

Die FLAMFILE selbst wird wie in der Vorg ngerversion aus formalen Gr nden satzweise geschrieben (z.B. fix 512 Bytes). Jeder Satz der FLAMFILE hat eine einfache Checksumme, mit der man sicherstellen will, dass es bei der bertragung nicht zu Formatfehlern gekommen ist. Das ist immer noch ein relativ h ufiger Anwenderfehler (v llig unabh ngig vom Einsatz von FLAM). Erst nach der Formatpr fung wird das Segment-Komprimat "zusammengebaut".

Jedes Segment-Komprimat hat einen Kopf. Dieser erm glicht es, in einer FLAMFILE zu positionieren (synchronisieren). Deshalb darf und wird er nicht verschleiert. Damit man aber sicher sein kann, dass die Informationen daraus korrekt sind, wird er separat ber eine Checksumme abgesichert.

Am Ende eines Segment-Komprimats findet man unseren Produktnamen FLAM in ASCII-Codierung. Dies erm glicht die Synchronisation bei Defekten oder beim Lesen von hinten.

Eine spezielle verdeckte Checksumme steht in direktem Zusammenhang mit der PASSWORD-Verschl sselung. Stimmt diese Checksumme nicht und ist das FLAG f r PASSWORD-Verschl sselung gesetzt, dann wurde versucht, mit einem falschen PASSWORD zu decodieren. Ist das PASSWORD-FLAG nicht gesetzt und benutzt jemand dennoch ein PASSWORD, wird ohne Hinweis auf diesen Eingabefehler decodiert und dekomprimiert.

Grunds tzlich beginnt die Dekomprimierung eines Segments nie, wenn irgendeine von den 4 Checksummen falsch ist. Dazu gibt es allein schon technische Gr nde. Die Dekomprimierung setzt eine gewisse sich st ndig ndernde Interpretation der Codierung voraus. Ein Defekt w rde dazu f hren, dass die Dekomprimierung unkontrolliert "aus dem Ruder" l uft. Das verhindert FLAM durch das Schichtenmodell mit 4 Checksummen. Wer dies trotz vorhandener Fehler (Fehlermeldungen, Return-Code) - etwa durch Manipulation mit Programmpatches - unterl uft, muss mit schwersten Folgefehlern rechnen.

Datenschutz und Datensicherheit, insbesondere Schutz vor unbefugten Angreifern hat - auch ohne PASSWORD-Verschl sselung - oberste Priorit t.

Das PASSWORD selbst darf 64 Bytes = 512 Bits lang sein. Man kann es abdruckbar mit C'...' oder hexadezimal

mit X'...' vorgeben. Bei der hexadezimalen Eingabe muss die Anzahl der quasi "halben" Bytes paarig aufgehen. Bei Eingabe mit C'...' muss man sich dessen bewusst sein, dass die bin re Umsetzung von der Systemgenerierung abh ngig ist. Das gleiche C-PASSWORD in Verbindung mit einer anderen Umsetzung der Zeichen in bin ren Code f hrt zu einem anderen internen PASSWORD. Das kann man als Vorteil nutzen, wenn man sich selbst in diesem Umfeld bewegt und nichts ndert. Die Abgrenzung mit Apostroph sichert, dass auch Blanks am Rand zum PASSWORD geh ren. Das PASSWORD mit C'...' muss exakt wiedergegeben werden, um decodieren zu k nnen. Es ist ratsam, bei jedem neuen PASSWORD beide Seiten vorab zu testen.

Bei falscher PASSWORD-Eingabe hat man auf Utility-Ebene genau einen Versuch, weil die interne bergabe innerhalb FLAM nur einen Versuch zul sst. F r einen weiteren Versuch muss man FLAM erneut starten und ein neues PASSWORD eingeben/zuweisen.

Das PASSWORD wird FLAM-intern so bearbeitet, dass es keine Chance gibt, R ckschl sse zu ziehen. Jeder Versuch einer Analyse, um sich einen Vorteil zu verschaffen, ist aussichtslos. Wir als Hersteller k nnen niemandem helfen, der sein PASSWORD vergisst. Es kann von aussen nicht einmal festgestellt werden, wie lang das benutzte PASSWORD war und ob es mit C'...' oder X'...' eingegeben worden ist. Hinweise von Hackern im Internet, wie man, um Zeit zu sparen, vorgehen sollte, wird man wohl kaum jemals finden.

Bevor das erste Segment einer FLAMFILE berhaupt entschl sselt werden kann, m ssen intern gewisse Vorbereitungsarbeiten ablaufen, die CPU-Zeit kosten und unumg nglich sind. Das bewirkt, dass man einen gewissen Mindestaufwand je PASSWORD-Versuch nicht optimieren kann. Die mathematisch nachvollziehbare Vielfalt an L sungen ist die sichere Garantie f r den Benutzer, ob jemand in vertretbarer Zeit ein zur Verschl sselung der FLAMFILE vorgegebenes PASSWORD "knackt". Ein bergeordnetes PASSWORD quasi als Universal-Schl ssel gibt es nicht. Ein aus Anwendersicht hierarchisch strukturiertes PASSWORD wird nicht als solches erkannt. Selbst der Unterschied von nur einem Blank mehr oder weniger am PASSWORD-Ende f hrt zu v llig unterschiedlichen internen Schl sseln, die allein ma geblich f r die tats chliche Vorgehensweise sind (2 * 4 KB Schl sseldaten intern).

Wenn Sie Ihrem PASSWORD immer noch ein Attribut geben, das sich auf Ihren Arbeitgeber oder Ihr sonstiges Umfeld bezieht und damit die PASSWORD-L nge k nstlich erweitern, dann steigt f r den Au enstehenden der Aufwand zur Ausforschung ins Astronomische:

Bei vollen 512 Bits bin r genutzt ergibt sich eine Anzahl von Varianten mit 155 Stellen. Selbst wenn nur je Byte 96 abdruckbare Zeichen zugelassen sein sollten, bleibt eine Zahl mit 127 Stellen. Allein die L nge, die PASSWORD-

Bestandteil ist, verunsichert, wenn man keine gezielten Informationen dazu hat.

Beispiel f r ein PASSWORD mit Attributen:

C'limes datentechnik gmbh, Zwiebackstadt Friedrichsdorf/Ts.'

Das sind 57 von 64 Bytes (zwischen den beiden Apostrophen). Alternativ zu "Zwiebackstadt" k nnte man als Attribute die Hugenotten, die Mormonen, Philipp Reis oder etwas anderes nehmen, das typisch f r Friedrichsdorf/Ts. ist. Den Rest (im Beispiel 7 Bytes) benutzt man f r das eigentliche individuelle PASSWORD (z.B. ein Blank und dann 6 Bytes variabler bin rer Code = 2,8 * 10**14 Varianten, wenn L nge, Aufbau und Attribut statisch sind).

Mit einem PASSWORD wie oben angegeben und ohne individuelle Modifikationen kann man sich ein "firmen-eigenes" FLAM-Komprimat erzeugen, das nur innerhalb der Firma dekomprimierbar ist. Dabei k nnte man statt "Ts." auch "Taunus" schreiben oder dieses Attribut ganz weglassen und durch die PLZ "D-61381" ersetzen. Gro - und Kleinschreibung beeinflussen die bin re Codierung ebenso wie ˜ nderungen im strukturellen Aufbau. Vorsicht bei Eingabefehlern im verdeckten Dialog und bei Kleinbuchstaben auf Mainframe.

Die PASSWORD-Verschl sselung kostet zus tzlich im Mittel 2,5% der Zeit f r die De-/Komprimierung mit FLAM V3.0 und MODE=ADC; allein durch die Beschr nkung auf komprimierte Daten ein immenser Vorteil. Letzteres gilt auch f r den Schutz vor Hackern, da zum "Angriff" der Besitz von FLAM V3.0 unerl sslich ist. Au erdem muss man jedes Segmentkomprimat vollst ndig und unversehrt in der richtigen H lle bereitstellen.

Unsere PC-Version ist keine Shareware o.dgl. und wir halten es f r nahezu ausgeschlossen, dass selbst nur die Dekomprimierung quasi in "fremder Eigenregie" nachprogrammiert und - wie im Internet blich - zum "Hausgebrauch" publiziert werden kann. Wir haben uns aus Selbstschutz um ein geh riges Ma an Komplexit t bem ht. Vor Raubkopien oder illoyalem Verhalten von Mitarbeitern, die Insider-Kenntnisse haben, kann man sich selbstverst ndlich nie sch tzen. Aber selbst damit k nnte man sich in gar keiner Weise irgendwelche Vorteile beim Versuch, ein PASSWORD zu "knacken", verschaffen. Der nicht optimierbare Aufwand an CPU-Zeit bleibt, selbst wenn wir die Sourcen ver ffentlichen! Diesen Aufwand bestimmen Sie durch Vorauswahl bei den PASSWORD-Vorgaben in Ihrem Hause (siehe PS). Bitte ber cksichtigen Sie, dass es einen gro en Unterschied macht, ob man sich im eigenen Hause oder "nur" vor "Unbefugten Dritten", z.B. beim File Transfer sch tzen will. Wer schon "im eigenen Haus" sitzt, hat meist noch ber andere Quellen Zugang zu den Daten, die Sie mit

viel Aufwand sch tzen wollen. Dieses Problem zu l sen, ist weitaus schwieriger.

Bei FLAM handelt es sich in der Regel um automatisierte Abl ufe. Wir w rden empfehlen, das PASSWORD in eine separate Datei zu legen und ber diese Datei von FLAM einlesen zu lassen. Der Zugriff auf die Datei l sst sich wie blich absichern.

In FLAM werden die f r die Synchronisation und Positionierung entscheidenden Teile der Syntax nicht verschl sselt und nicht verschleiert. Mit diesen Daten kann niemand etwas anfangen; sie k nnen aber dazu beitragen, den direkten Zugriff enorm zu beschleunigen, weil die Teile des Komprimats, die den berechtigten Anwender interessieren, weder entschl sselt noch entschleiert und nicht unn tig dekomprimiert werden m ssen.

Selbstverst ndlich kann jeder Anwender auch den Weg gehen, dass erst mit FLAM komprimiert und verschleiert wird, und danach benutzt man ein vorgeschriebenes Verschl sselungsverfahren. Die Originaldaten vor der Komprimierung mit FLAM zu verschl sseln, bringt hingegen nichts. Man kann aber durchaus Signaturen und andere Daten zur Autorisierung ber das Original bilden, ehe man mit FLAM komprimiert, wenn dabei die Daten im Original im Prinzip nicht ver ndern werden.

Anstelle individueller Schl ssel kann man fertige Schl sselsysteme mit Generierung/Verwaltung etc. benutzen, nur m ssen die Schl ssel bei der FLAM-Verschl sselung symmetrisch sein (auf beiden Seiten das gleiche PASSWORD aus bin rer Sicht).

PS: Wenn Sie sich ausrechnen wollen, wieviel PASSWORD-Varianten es gibt, dann m ssen Sie bei rein bin ren Codes (X-Eingabe) die L nge in Bits als Potenz zur Basis "2" nehmen, wobei es eine Zahl sein muss, die ohne Rest durch "8" teilbar ist, die Eingabel nge geht auf volle Bytes. Im X-Format ist das PASSWORD bei heterogen Anwendungen in je Fall invariant.

Bei Eingabe mit C’...’ kommt es darauf an, wieviel Zeichen erlaubt sind. Es gibt z.B. in ASCII 96 abdruckbare Zeichen (ausgenommen erweiterte Zeichens tze). Davon sind nur 52 Zeichen lateinische Buchstaben etc. pp.. Hat das PASSWORD eine L nge von "k" Bytes und gibt es je Byte max. "n" Zeichen, die zul ssig sind, dann betr gt die Menge an Variationen "n**k" (Potenz "k" zur Basis "n"). Es gibt immer einen "Bodensatz", den ein Angreifer ausschlie en wird. Deshalb ist es schon wichtig, in der gew hlten L nge "k" genug "Luft" zu lassen (vgl. das Beispiel mit PASSWORD-Attributen). Das C-PASSWORD ist von Zeichens tzen und deren bin rer Umsetzung ggf. extrem abh ngig, z.B. bei Sonderzeichen und Umlauten! F r FLAM ist allein die bin re Umsetzung des beim Komprimieren und Verschl sseln mit C’...’ bergebenen

Strings g ltig. Das kann schon am n chsten Bildschirm eine andere bin re Codierung sein.

1.2 FLAM und AES

Der ’Advanced Encryption Standard’ (AES) l st den in die Jahre gekommenen ’Data Encryption Standard’ (DES) ab.

Dieser moderne symmetrische Blockalgorithmus bildet die Basis f r die kryptographische Absicherung einer FLAMFILEfi ab FLAMfi 4.0.

Er ist gegen ber DES wesentlich sicherer und ben tigt gleichzeitig nur ein Zehntel der Rechenzeit. Dies - in Verbindung mit der ADC-Komprimierung - macht es m glich, starke Kryptographie auf gro e Datenmengen anzuwenden.

In FLAMfi wird AES mit einer Block- und Schl ssell nge von jeweils 128 Bits (16 Bytes) eingesetzt.

Die Verschl sselung mit AES wird von FLAMfi im MODE=ADCfi (Advanced Data Compression) oder im MODE=NDC (No Data Compression) – einer Unterfunktion der ADC-Algorithmik – unterst tzt. Mit NDC werden die reinen Nettodaten nur 1:1 kopiert. Damit kann auch jede FLAMFILEfi im "Nachhinein" ohne Performance-Verluste (2-Schritt-Verfahren) verschl sselt werden. Auf diese Weise kann sogar eine "leere" Datei so verschl sselt werden, dass "leer" nicht mehr erkennbar ist.

Die Vertraulichkeit und Integrit t einer FLAMFILE wird mit sogenannten Hash-MACs sichergestellt.

Bei diesem Schutz handelt es sich um reine Software-Kryptographie, was bedeutet, dass die verwendeten Schl ssel - wenn auch nur kurzzeitig - in klarer Form auf dem Rechner, wo die FLAMFILEfi erzeugt wird, vorkommen. Da aber zu diesem Zeitpunkt auch die Originaldaten auf diesem Rechner existieren, kann ein Angreifer, der Zugriff auf den Rechner erlangt hat, gleich die klaren Daten aussp hen. Der verwendete Schl ssel nutzt ihm nur etwas, wenn dieser erneut zur Anwendung kommt und der Angreifer dann keinen Zugriff mehr auf das System hat.

Die maximale Sicherheit, die FLAMfi mit AES bieten kann, ist abh ngig von der *Sicherheit der Rechner*, auf denen die FLAMFILEfi geschrieben bzw. gelesen wird. FLAMfi stellt mit AES kryptographisch sicher, dass auf dem bertragungsweg niemand ohne die Kenntnis des Schl ssels Daten manipulieren oder aussp hen kann. Man kann diese Sicherheit noch verbessern, indem man die verschl sselte FLAMFILEfi zwischen Servern austauscht, auf denen weder FLAMfi noch die Originaldaten verf gbar sind. Dies ist eine einfache organisatorische Ma nahme, die die Sicherheit wesentlich erh ht. Diese organisatorische L sung mit FLAMfi ist

auch wesentlich sicherer als eine Kombination aus File Transfer und integrierte Kryptographie in direkter Verbindung zwischen Sende- und Empfangssystem.

Kryptographie allein - ohne ein angepasstes organisatorisches Umfeld - ist kein Garant f r Sicherheit.

Eine in Verbindung mit Kryptographie organisatorisch interessante L sung, die FLAMfi bietet, ist das Parallel-Splitting. Durch die gleichm ige Verteilung der verschl sselten FLAMFILEfi in Einheiten von nur 4 Bytes parallel auf mehrere Teildateien, kann man nur decodieren, wenn man den Schl ssel und alle zusammengeh renden Teildateien gleichzeitig an FLAMfi   bergibt. Damit kann u.U. das Problem der Synchronisation des Schl ssels gel st werden (z.B. in der Langzeit-Archivierung durch Verteilung auf verschiedene Standorte).

Es gibt in FLAMfi seit V4.0 ein Feature, mit dem man eine FLAMFILEfi - ob verschl sselt oder nicht - auf ihre technische Integrit t pr fen kann (Checksummen auf der Basis von CRC-Routinen). Solche Techniken sind z.B. in Verbindung mit File-Transfer international allgemeiner Standard. *Sie sch tzen nicht vor Manipulation.*

Unabh ngig davon kann man eine mit FLAMfi V4 und AES verschl sselte FLAMFILEfi - *ohne zu dekomprimieren* - auf ihre Integrit t gem   den Anforderungen der Kryptographie pr fen. Dazu muss man allerdings den Schl ssel benutzen, mit dem diese FLAMFILEfi erzeugt worden ist.

Weitergehende Informationen, insbesondere zur Arbeitsweise von FLAM mit AES, entnehmen Sie bitte dem Handbuch *FLAM & AES*, das jeder Auslieferung beigef gt ist.

FLAM (VSE)

Benutzerhandbuch

Kapitel 2:

Funktionen

Inhalt

2. Funktionen

2.1 Dienstprogramm FLAM

Das Dienstprogramm FLAM kann ganze Dateien komprimieren oder komprimierte Dateien expandieren.

Mit den Parametern COMPRESS bzw. UNCOMPRESS / DECOMPRESS kann bestimmt werden, ob eine Originaldatei komprimiert oder eine Komprimatsdatei expandiert werden soll.

2.1.1 Komprimieren von Dateien

FLAM komprimiert eine Datei und schreibt das Ergebnis, die FLAMFILE, als sequentielle oder indexsequentielle Datei. In dieser FLAMFILE k nnen in einem Header Informationen ber den originalen Datenbestand gespeichert werden.

FLAM kann alle DTFSD-,DTFMT-und alle VSAM-Dateien verarbeiten. Seit der FLAM Version 3.0 k nnen auch MEMBER aus LIBR-Bibliotheken und POWER-QUEUES verarbeitet werden.

Um die Komprimierung auf die Erfordernisse des Anwenders einzustellen, k nnen Parameter beim Aufruf des Programms im Dialog vorgegeben werden. Die Parameter k nnen auch ber eine Parameterdatei und durch Generierung eingestellt werden.

FLAM protokolliert den Verarbeitungsablauf in eine Meldungsdatei.

Bei der Komprimierung mit FLAM werden 1-255 (logische) S tze in einem Block (Matrix) zusammen bearbeitet.

Dateien k nnen von der Platte und direkt vom Magnetband gelesen bzw. geschrieben werden. Dies gilt auch f r die FLAMFILE selbst.

Magnetbanddateien m ssen mit JCL (TLBL-Anweisung) zugeordnet werden.

Grunds tzlich komprimiert FLAM mehrere Datens tze zusammen. Der Zwischenpuffer kann bei der Komprimierung mit CX8/VR8 mit dem MAXBUFFER-Parameter dimensioniert werden. Im ADC/NDC-Mode werden zwei Puffer zu 64 KB angelegt. Es werden nur so viele Datens tze eingelesen wie vollst ndig zwischengespeichert werden k nnen.

Mit dem MAXRECORDS-Parameter kann die Satzanzahl limitiert werden. Bei MAXRECORDS=1 findet eine serielle, kontextfreie Komprimierung statt, die nur bei l ngeren Datens tzen sinnvoll ist.

Sind Dateien unstrukturiert, dann ist MODE=ADC die geeignete Komprimierungsvariante. Der Parameter MAXRE-CORDS sollte auf 4095 eingestellt werden (geschieht automatisch ohne Angabe).

Die verfahrenstypische Komprimierung ist bereits bei 16-32 Datens tzen je Matrix effizient. H here Blockungen verbessern zwar den Komprimierungseffekt und f hren damit zu einem geringeren CPU-Zeitverbrauch, ben tigen andererseits aber gr ere Zwischenpuffer. Je schlechter der Komprimierungseffekt ist, desto mehr CPU-Zeit wird verbraucht.

Die Komprimierungstechnik ist im Prinzip immer gleich, sie basiert auf dem Frankenstein-Limes-Verfahren. Nur in der Behandlung der Matrix-Spalten und der Darstellung des Komprimats gibt es Unterschiede, die ber den MODE-Parameter gesteuert werden.

Mit CX8 werden nur Zeichenwiederholungen komprimiert, w hrend mit VR8 die verbleibenden Reste nach dem FL-B(4)-Code nachkomprimiert werden. Dabei werden die Zeichen zun chst in einen speziellen 8-Bit-Code bersetzt und in diesem durch logische Operationen homogenisiert. Dadurch entstehen Bitketten, die sich effizient komprimieren lassen, zumal die Reste aufgrund vertikaler Vorgehensweise partiell gleichen Zeichenklassen angeh ren.

Das Komprimat, die FLAMFILE, ist in beiden F llen eine Folge von beliebigen 8-Bit-Kombinationen, die als sequentielle oder indexsequentielle Datei weggeschrieben wird. Satzl nge, Satzformat und Blockgr e kann der Anwender selbst bestimmen. Jeder Satz dieser Datei wird durch eine Checksumme vor Datenverf lschung gesch tzt. Codekonvertierungen im Komprimat sind unzul ssig. Die Datei ist bei bertragungen wie eine Bin rdatei zu behandeln.

F r Dateien, die nur aus abdruckbaren Zeichen bestehen und die ber eine 7-Bit-Leitung transportiert werden sollen, bietet FLAM den MODE=CX7 an. Dieser erzeugt ein Komprimat, das sich in Bezug auf die bertragung nicht anders als die Original-Datei selbst verh lt. Eine Pr fung hinsichtlich der " bertragbarkeit" erfolgt nicht. FLAM selbst benutzt zur Darstellung des Komprimats einen stark eingeschr nkten Zeichenvorrat, der sich invariant zu marktg ngigen Konvertierungen verh lt.

In diesem Modus ist es also zul ssig, das Komprimat von EBCDIC nach ASCII oder umgekehrt zu konvertieren (z. B. w hrend eines Filetransfers).

Entscheidend ist, dass solche Konvertierungen exakt 1:1 ablaufen m ssen. FLAM moniert sonst beim Dekomprimieren Syntax-Fehler wegen Abweichungen in der Byte-Anzahl und bricht ab. Solche F lle sind denkbar, wenn z. B. Steuerzeichen in Druckdateien oder Tabulatorzeichen nicht 1:1 konvertiert werden.

Unabh ngig davon, bietet FLAM dem Anwender die M glichkeit, jeden Datensatz vor der Komprimierung und/oder nach der Dekomprimierung zeichenweise ber Standardtabellen oder benutzereigene Tabellen konvertieren zu lassen. F r Konvertierungen, die nicht 1:1 ber alle Zeichen erfolgen d rfen, k nnen Benutzerausg nge verwendet werden.

2.1.2 Dekomprimieren von Dateien

FLAM liest eine komprimierte Datei (FLAMFILE), dekomprimiert den Inhalt und gibt die dekomprimierten Daten in eine Datei aus. Es erkennt dabei selbst ndig mit welchen Parametern (wie Puffergr e oder max. Satzanzahl) die FLAMFILE erzeugt worden ist. Der Aufbau der Komprimatsdatei wird in einem eigenen Kapitel beschrieben .

FLAM in der Version 3.0 kann alle Komprimatsdateien der Vorg ngerversion dekomprimieren (Aufw rtskompatibili-t t).

Um die Dekomprimierung auf die Erfordernisse des Anwenders einzustellen, k nnen Parameter vorgegeben werden. Die Parameter k nnen auch ber eine Parameterdatei und durch Generierung eingestellt werden.

FLAM protokolliert den Ablauf wahlweise am Bildschirm oder in eine Meldungsdatei. Bei der Dekomprimierung werden die Kenndaten der Originaldatei wieder hergestellt, soweit diese in einem Fileheader zur Verf gung stehen.

Durch Parameterangaben f r die Ausgabedatei ist es beispielsweise m glich, bestimmte Kenndaten zu ndern. Alle Konvertierungen sind m glich und erlaubt, vorausgesetzt FLAM unterst tzt die entsprechende Zugriffsmethode des Datenverwaltungssystems.

Stammt das Komprimat (die FLAMFILE) von einem anderen Betriebssystem, so ndert das an dem Verhalten von FLAM nichts. Die Daten werden in quivalente Dateien dekomprimiert oder k nnen gegebenenfalls in ein vom Anwender vorgegebenes Format umgesetzt werden.

Durch Angabe von bersetzungstabellen ist FLAM in der Lage, Daten nach der Dekomprimierung gem dieser Tabelle umzuschl sseln.

Um eine weitgehende Flexibilit t zu erreichen, kann ein Benutzerausgang aktiviert werden, der die Daten nach der Dekomprimierung in gew nschter Weise bearbeitet.

2.2 Unterprogramm FLAMUP

FLAMUP unterscheidet sich von FLAM nur dadurch, dass es als Unterprogramm aufgerufen werden kann. Alle Zugriffe auf die Datenbest nde werden weiterhin von FLAM-Modulen bernommen.

Die Parameter k nnen bei Aufruf bergeben werden und/oder wie beim Dienstprogramm vom Bildschirm oder aus einer Parameterdatei gelesen werden.

Mit FLAMUP ist es beispielsweise m glich, ber ein Rahmenprogramm eine definierte Menge von Dateien zu selektieren und innerhalb des Programmlaufs automatisch zu komprimieren /dekomprimieren. Die Selektion k nnte z.B. alle Dateien umfassen, die ab einem bestimmten Zeitpunkt ge ndert wurden (Archivierung).

2.3 Satzschnittstelle FLAMREC

Die Frankenstein-Limes-Zugriffsmethode wird durch die Satzschnittstelle als Hersteller unabh ngige, komprimierende Dateizugriffsmethode realisiert.

Sie erm glicht den sequentiellen, relativen und indexsequentiellen Zugriff auf einzelne Originals tze von Komprimaten, die auf unterschiedlichen Datentr gern verschiedener Betriebssysteme abgelegt und zwischen diesen ausgetauscht werden k nnen.

Die Satzschnittstelle wird durch eine Reihe von Unterprogrammen dargestellt, die von allen Programmiersprachen wie COBOL, FORTRAN, C und ASSEMBLER aufgerufen werden k nnen.

Diese Unterprogramme sind auf allen Betriebssystemen, f r die FLAM ab der Version 2.5 verf gbar ist, gleich bzw. quivalent.

FLMCLS

FLMCLS (Close) schlie t die Verarbeitung ab, nachdem alle S tze an FLAM bergeben, oder beim Dekomprimieren alle Originals tze gelesen wurden.

FLMDEL

FLMDEL (Delete) l scht den zuletzt gelesenen Satz aus einer indexsequentiellen FLAMFILE.

FLMFKY

Mit FLMFKY (Find Key) wird in einer indexsequentiellen FLAMFILE, die aus einer indexsequentiellen Datei erzeugt wurde, so positioniert, dass mit einem anschlie enden FLMGET der Satz mit dem vorgegebenen oder dem folgenden Schl ssel gelesen werden kann.

FLMFLU

Mit FLMFLU (Flush) wird evtl. noch im Speicher befindliches Komprimat der zuletzt zur Komprimierung bergebenen S tze in die FLAMFILE ausgegeben und die Statistikdaten angefordert. Im Gegensatz zu FLMCLS wird

die FLAMFILE nicht geschlossen, d.h. ein weiteres Komprimat kann angef gt werden.

FLMFRN

Mit FLMFRN (Find Record Number) wird in einer indexsequentiellen FLAMFILE, die aus einer relativen oder sequentiellen Datei erzeugt wurde, so positioniert, dass mit einem anschlie enden FLMGET der Satz mit der vorgegebenen Satznummer gelesen werden kann.

FLMGET

FLMGET (Get Record) liest einen dekomprimierten Originalsatz in einem vorgegebenen Puffer.

FLMGHD

Mit FLMGHD (Get Fileheader) kann die Fileheaderinformation ber die Originaldatei gelesen werden. Falls mehrere Fileheader in der FLAMFILE vorhanden sind, beziehen sich diese Informationen auf die Originals tze, die mit den Funktionen FLMGET, FLMLOC als n chste gelesen werden.

FLMGKY

Mit FLMGKY (Get Key) kann ber einen Schl ssel ein Satz aus einer FLAMFILE von einem indexsequentiellen Original gelesen werden. Dabei wird gleichzeitig f r das sequentielle Lesen mit FLMGET bzw. FLMLOC auf den Satz mit dem n chst gr eren Schl ssel positioniert.

FLMGRN

Mit FLMGRN (Get Record Number) wird aus einer index-sequentiellen FLAMFILE, die aus einer relativen oder sequentiellen Datei erzeugt wurde, der Satz mit der vorgegebenen Satznummer gelesen.

FLMGTR

FLMGTR (Get Reverse) liest den n chsten dekomprimierten Originalsatz in Richtung auf den Dateianfang in einen vorgegebenen Puffer.

FLMGUH

Informationen, die bei der Komprimierung mit FLMPUH in das Komprimat eingef gt wurden, k nnen bei der Dekomprimierung mit FLMGUH (Get User Header) gelesen werden.

FLMIKY

Mit FLMIKY (Insert Key) wird ein Satz mit neuem Schl ssel in das Komprimat bernommen. Der angegeben Schl ssel darf noch nicht in der Datei existieren.

FLMLCR

FLMLCR (Locate Reverse) liest den n chsten dekomprimierten Originalsatz in Richtung auf den Dateianfang im Locate Mode.

FLMLOC

Anstelle von FLMGET kann auch die Funktion FLMLOC (Locate Record) verwendet werden. Dabei wird jedoch kein Satz in den Puffer bertragen, sondern es wird lediglich die Adresse dieses Satzes zur ckgegeben.

FLMOPN

Die Funktion FLMOPN (Open) ist aufgrund der gro en Anzahl von Parametern in die drei Teilfunktionen FLMOPN, FLMOPD und FLMOPF untergliedert worden. FLMOPN gibt die wichtigsten Parameter (z. B. komprimieren oder dekomprimieren) an FLAM weiter. Mit der Funktion FLMOPD werden die Dateieigenschaften der

FLAMFILE festgelegt, und FLMOPF bestimmt die Komprimatseigenschaften. Kommen die Teilfunktionen FLMOPD und FLMOPF nicht zur Anwendung, so werden feste Werte verwendet.

FLMPHD

Mit der Funktion FLMPHD (Put Fileheader) k nnen beim Komprimieren die Dateieigenschaften der Originals tze beschrieben werden, damit diese Eigenschaften im Fileheader ablegt werden. Der Fileheader gilt dabei f r die anschlie end mit FLMPUT bergebenen Originals tze.

FLMPKY

Mit FLMPKY (Put Key) kann ein Satz mit angegebenem Schl ssel in einer indexsequentiellen FLAMFILE ge ndert oder eingef gt werden.

FLMPOS

FLMPOS (Position) dient zum relativen Positionieren in beliebigen Dateien und beim Schreiben von relativen Dateien zum Erzeugen von L cken.

FLMPUH

An die mit FLMPHD gespeicherten Informationen kann mit der Funktion FLMPUH (Put User Header) noch eine Zeichenkette beliebigen Inhalts angef gt werden. Der Aufruf darf nur unmittelbar nach einem FLMPHD-Aufruf erfolgen.

FLMPUT

FLMPUT (Put Record) bergibt einen Originalsatz zum komprimieren an FLAM.

FLMPWD

FLMPWD bergibt ein Passwort zur Komprimierung bzw. Dekomprimierung an FLAM.

FLMQRY

FLMQRY erfragt Parameterwerte, die FLAM aktuell verwendet.

FLMSET

FLMSET setzt Parameter f r den Ablauf mit FLAM.

FLMUPD

Mit FLMUPD (Update) wird der jeweils zuletzt gelesene Originalsatz aus einer indexsequentiellen FLAMFILE ge ndert.

2.4 Benutzer Ein-/Ausgabe Schnittstelle

Mit dieser Schnittstelle k nnen eigene Zugriffsfunktionen in FLAM integriert werden.

So k nnen beispielsweise die Komprimatss tze unmittelbar weiter verarbeitet werden, ohne dass zun chst eine Datei erzeugt werden muss, bzw. Komprimatss tze k nnen unmittelbar bernommen werden.

Eine praktische Anwendung dieses Konzeptes erm glicht die Integration von FLAM mit einem Filetransfer ohne den Umweg ber Zwischendateien.

ber diese Schnittstelle k nnen aber auch die Eingabe- und Ausgabedaten des Dienstprogramms FLAM oder des Unterprogramms FLAMUP bearbeitet werden. Hier kann FLAM mit geringem Aufwand an spezielle Zugriffsverfahren angepasst werden.

2.5 Benutzerausg nge

Die Benutzerausg nge werden aus der FLAM-Lade-bibliothek geladen.

2.5.1 Eingabe Originaldaten EXK10

Von diesem Benutzerausgang wird der zu komprimierende Satz unmittelbar nach dem Lesen aus der Eingabedatei zur Verf gung gestellt.

Hier k nnen Verarbeitungen am Dateianfang, bei jedem Satz und am Dateiende durchgef hrt werden. Es k nnen S tze bernommen, ver ndert, gel scht und eingef gt werden. Dieser Benutzerausgang ist geeignet, S tze strukturorientiert zu ver ndern.

EXK10 ist nur in FLAM und FLAMUP verf gbar und korrespondiert mit EXD10 bei der Dekomprimierung.

2.5.2 Ausgabe Komprimat EXK20

Von diesem Benutzerausgang wird das Komprimat zur Verf gung gestellt, unmittelbar bevor es in die FLAMFILE geschrieben wird.

Es k nnen Verarbeitungen am Dateianfang, bei jedem Satz und am Dateiende durchgef hrt werden. Dieser Benutzerausgang ist geeignet, S tze Struktur unabh ngig zu bearbeiten.

Hier kann z.B. das Komprimat mit einer eigenen Verschl sselungsroutine bearbeitet werden, oder es kann eine Code-Umsetzung vorgenommen werden, um eine nicht transparente Daten bertragung nutzen zu k nnen. Es lassen sich S tze vor dem Komprimat einf gen, um z.B. eigene Archivierungsdaten oder Herkunftsangaben zu speichern.

Eine weitere M glichkeit liegt in der Verl ngerung von Datens tzen, um bestimmte revisionsspezifische Daten aufzunehmen.

EXK20 ist in FLAM, FLAMUP und FLAMREC verf gbar und korrespondiert mit EXD20 bei der Dekomprimierung.

2.5.3 Ausgabe Originaldaten EXD10

In diesem Benutzerausgang wird der dekomprimierte Satz unmittelbar vor dem Schreiben in die Ausgabedatei zur Verf gung gestellt.

In diesem Benutzerausgang k nnen Verarbeitungen am Dateianfang, bei jedem Satz und am Dateiende durchgef hrt werden. Es k nnen S tze bernommen, ver ndert, gel scht und eingef gt werden. Dieser Benutzerausgang ist geeignet, S tze strukturorientiert zu bearbeiten.

EXD10 ist nur in FLAM und FLAMUP verf gbar und korrespondiert mit EXK10 bei der Komprimierung.

2.5.4 Eingabe Komprimat EXD20

In diesem Benutzerausgang wird das Komprimat unmittelbar nach dem Lesen aus der FLAMFILE zur Verf gung gestellt.

In diesem Benutzerausgang k nnen Verarbeitungen am Dateianfang, bei jedem Satz und am Dateiende durchgef hrt werden. Dieser Benutzerausgang ist geeignet, S tze strukturunabh ngig zu bearbeiten.

Hier kann beispielsweise das Komprimat entschl sselt oder eine eigene Code-Umsetzung wegen der Daten bertragung r ckg ngig gemacht werden.

Zur fehlerfreien Arbeitsweise von FLAM ist es absolut notwendig, dass alle ˜nderungen am Komprimat reversibel sind. Am Ende des Benutzerausgangs EXD20 m ssen die gleichen Daten bereitgestellt werden, die dem Benutzerausgang EXK20 am Eingang bergeben worden sind. Alle durch EXK20 erzeugten Ver nderungen sind in EXD20 r ckg ngig zu machen.

EXD20 ist in FLAM, FLAMUP und FLAMREC verf gbar und korrespondiert mit EXK20 bei der Komprimierung.

2.5.5 Schl sselverwaltung KMEXIT

Durch diese Benutzerroutine wird dem Dienstprogramm FLAM ein Schl ssel zur Ver-/Entschl sselung zur Verf gung gestellt.

Damit ist der Anschluss an eine Schl sselverwaltung unabh ngig von FLAM m glich. Die verwendeten Schl ssel werden nicht protokolliert und treten somit nach au en nicht in Erscheinung.

FLAM (VSE)

Benutzerhandbuch

Kapitel 3:

Schnittstellen

Inhalt

3. Schnittstellen

FLAM bietet eine Reihe von Schnittstellen, die es erm glichen, das Produkt in unterschiedlichen Umgebungen und f r verschiedene Aufgaben einzusetzen.

Die einfachste Anwendung ist der Aufruf ber das EXEC-Kommando. Damit k nnen vollst ndige Dateien komprimiert bzw. dekomprimiert werden.

Daneben bietet FLAM eine Reihe von Unterprogramm-Schnittstellen, die die Integration mit anderen Programmen und Produkten erm glichen. Weiterhin k nnen damit ma geschneiderte Anwendungen entwickelt werden, indem FLAM in Steuerungsprogramme eingeh ngt wird.

Benutzerausg nge erm glichen die Vor- und Nachbearbeitung der Originaldaten und Komprimate, ohne den Umweg ber Zwischendateien.

Alle Schnittstellen sind so ausgelegt, dass eine Benutzung von h heren Programmiersprachen wie COBOL m glich ist. Nur wenn die Verwendung von Pointern unvermeidbar ist, muss die Schnittstelle in ASSEMBLER o. . genutzt werden.

3.1 Dienstprogramm FLAM

Mit FLAM k nnen vollst ndige Dateien komprimiert und Komprimate wieder in vollst ndige Dateien rekonstruiert werden.

Als Originaldateien sind alle VSAM-, DTFSD-, DTFMT-Dateien und MEMBER aus LIBR-Bibliotheken oder POWER-QUEUES zugelassen.

ber die Benutzerschnittstelle f r den Dateizugriff (DEVICE=USER) ist es m glich, weitere Zugriffsmethoden zu unterst tzen.

Sowohl die Originaldaten als auch die Komprimate k nnen an Benutzerausg ngen auf einfache Art vor- bzw. nachbearbeitet werden. Dabei sind Benutzerausg nge Unterprogramme, die zur Laufzeit dynamisch aus einer VSE-LIBR-Bibliothek (LIBDEF) nachgeladen werden.

Die Originaldaten k nnen mit Hilfe von fest definierten und dynamisch ladbaren bersetzungstabellen zeichenweise umcodiert werden.

Beim Dekomprimieren k nnen die Datei- und Satzformate konvertiert werden. Dabei sind z.B. Umwandlungen von variablem in fixes Format oder von indexsequentieller in sequentielle Organisation m glich.

Die Komprimate k nnen in sequentiellen Dateien mit beliebigen Satzformaten abgelegt werden. Das Satzformat f r die Komprimate ist unabh ngig vom Satzformat der Originaldateien.

FLAM Komprimate sind immer heterogen kompatibel. Das hei t Komprimate, die unter einem Betriebssystem erzeugt wurden, k nnen immer auf allen anderen Betriebssystemen dekomprimiert werden, f r die FLAM verf gbar ist. Gegebenenfalls m ssen dabei die Satz- und Dateiformate beim Dekomprimieren konvertiert werden.

FLAM kann sehr flexibel an die Erfordernisse des Benutzers angepasst werden. Dabei sind verschiedene Mechanismen f r die Parametrisierung vorgesehen.

Die Parameter k nnen ber die PARM-Schnittstelle gelesen werden. Au erdem ist das Einlesen aus einer Parameterdatei und JCL-Stream vor /*-Karte vorgesehen. Und zus tzlich k nnen die Parameter durch Generierung fest eingestellt werden (siehe: Standardwerte generieren).

Bei der Verarbeitung werden die Parameter in folgender Reihenfolge ausgewertet:

Zun chst werden die Parameter aus der Generierung genommen. Bei der Dekomprimierung werden diese Parameter von den im Fileheader gespeicherten Werten berschrieben, sofern dieser vorhanden ist.

Danach werden die Werte aus der Parameterdatei gelesen.

Die PARM-Eingabe berschreibt ihrerseits wieder die Angaben aus der Parameterdatei.

Durch diese Hierarchie ist eine sehr flexible Bedienung m glich. Es ist zu beachten, dass die Reihenfolge nicht immer chronologisch ist:

Es ist beispielsweise m glich, in der PARM-Eingabe die Parameterdatei auszuw hlen, die erst nach dem Ende der Eingabe eingelesen wird, obwohl die PARM-Eingaben die Angaben in der Parameterdatei berschreiben.

3.1.1 Parameter

Unabh ngig vom Eingabemedium werden die Parameter nach der gleichen Syntax interpretiert. Es d rfen nur Gro buchstaben benutzt werden. Die Parameter k nnen in einer oder mehreren Zeilen bzw. S tzen bergeben werden. In jeder Zeile endet die Interpretation des Parameterstrings mit dem ersten Leerzeichen. Danach kann ein beliebiger Kommentar folgen. Einzelne Parameter d rfen nicht durch Zeilenenden getrennt werden. Die Verarbeitung der Parameter endet durch das Schl sselwort "END" bzw. durch eine leere Eingabe (L nge=0) oder EOF f r das Eingabemedium.

Es gibt Parameter mit oder ohne Schl sselworte. Die Schl sselworte und Werte k nnen abgek rzt werden. In eckigen Klammern [ ] sind alternative Schl sselworte dargestellt, die in anderen Betriebssystemen gebr uchlich sind (MVS, VM, BS2000).

Aus Kompatibilit tsgr nden sind alle Parameter beschrieben, obwohl einige der Parameter unter VSE nicht ausgewertet werden.

Die Schl sselwortparameter k nnen in zwei Schreibweisen angegeben werden:

parameter0,parameter1=wert1,parameter2=wert2, ...

oder auch:

parameter0,parameter1(wert1),parameter2(wert2),. ..

Alle Parameter, die Zeichenfolgen aufnehmen (Dateinamen, Modulnamen usw.), werden mit Leerzeichen gef llt, wenn "(NONE)" oder gar kein Wert angegeben wird:

```
parameter=(NONE),   bzw.  parameter(NONE),...
```

oder auch:

```
parameter=,...  bzw.    parameter(),...
```

F r Zeichenfolgen sind drei Schreibweisen zul ssig. Eine abdruckbare Zeichenfolge kann direkt angegeben werden:

```
FLAMIN=VSESPUC:U.LST bzw. FLAMIN(VSESPUC:U.LST)
```

Sie kann als abdruckbare Zeichenfolge gekennzeichnet werden:

FLAMIN=C'VSESPUC:U.LST' bzw.
FLAMIN(C'VSESPUC:U.LST')

Zeichenfolgen k nnen aber auch in hexadezimaler Darstellung eingegeben werden:

```
FLAMIN=X'E5E2C5E2D7E4C37AE44BD3E2E3'   bzw.

FLAMIN(X'E5E2C5E2D7E4C37AE44BD3E2E3')
```

Die Reihenfolge der Parameter ist beliebig, sofern nicht anders beschrieben.

Es m ssen nur Parameter, die von den Standardwerten abweichen, angegeben werden.

Im folgenden sind alle Parameter in alphabetischer Reihenfolge aufgef hrt und beschrieben.

Die Parameter k nnen abgek rzt werden, solange sie eindeutig bleiben. Andernfalls wird der erste bereinstimmende Eintrag gem der folgenden bersicht genommen.

ACCESS

ACC

Zugriffsverfahren auf die Eingabe- bzw. Ausgabedatei.

M gliche Werte:

LOG	logisch satzweiser Zugriff
PHY	physischer blockweiser Zugriff
MIX	physischer Zugriff mit logischer Entblockung
Standard:	LOG
G ltig f r:	Komprimierung, Dekomprimierung

Hinweis: PHY ist nur bei "No Label Tapes" erlaubt.
Sonst werden alle Dateien logisch gelesen und geschrieben.

BLKSIZE

BLKS

Logische Blockl nge f r die Komprimatsdatei.

M gliche Werte:

0 - 32760

Standard:	6144 Bytes
G ltig f r:	Komprimierung, Dekomprimierung

BLKMODE

BLKM

Blockmodus f r sequentielle Komprimatsausgabe.

M gliche Werte:

YES	In einem Komprimatssatz k nnen Daten aus mehreren Matrizen vorhanden sein.
NO	In einem Komprimatssatz befinden sich nur Daten aus einer Matrix. Bei Matrixwechsel beginnt jeweils ein neuer Komprimatssatz.
Standard:	YES
G ltig f r:	Komprimierung

Hinweis: Dieser Parameter ist nur aus Kompatibilit ts-gr nden zur Version 2.0 eingef hrt, da sequentielle Komprimatsdateien mit fixer Satzl nge nur bei BLKMODE= YES verarbeitet werden k nnen. F r den Direktzugriff ist BLKMODE=NO Voraussetzung.

CHECKALL

CHECKA

Komplette Pr fung einer FLAMFILE einschlie lich der Dekomprimierung und ggf. Entschl sselung, aber ohne Dateiausgabe.

Keine Werte

g ltig f r:	Dekomprimierung

Hinweis: wurde die FLAMFILE verschl sselt, so ist der Schl ssel anzugeben.

Der Parameter CHECKALL ist eine Kurzform f r
DECOMPRESS,FLAMOUT=*DUMMY,SHOW=ALL

CHECKFAST

CHECKF

Pr fung einer FLAMFILE auf Integrit t und Vollst ndigkeit ohne Dekomprimierung aber ggf. mit Entschl sselung

Keine Werte

g ltig f r:	Dekomprimierung

Hinweis: Kann z.B. zur Pr fung nach File Transfers verwendet werden. Mit Angabe des Schl ssels wird zus tzlich die Entschl sselung durchgef hrt und es werden alle MACs gepr ft.

Der Parameter CHECKFAST ist eine Kurzform f r DECOMPRESS,SHOW=DIR

COMPRESS

C

Komprimieren

keine Werte

g ltig f r: Komprimierung

CLIMIT

CLI

Minimale Komprimierung in Prozenten.

M gliche Werte:

0 - 90

Standard: 0 kein Grenzwert

g ltig f r: Komprimierung

Hinweis: Wird die Komprimierung schlechter als der vorgegebene Grenzwert, so wird von FLAM eine Meldung erzeugt und der Returncode 4 gesetzt.

Die Komprimierung wird trotzdem ordnungsgem zu Ende gef hrt. Dieser Parameter wird nur bei INFO=YES bzw. bei SHOW=ALL ausgewertet.

CLOSDISP

CLO

Endeverarbeitung f r Komprimatsdatei auf Band.

M gliche Werte:

REWIND	Zur ckspulen des Bandes an den Anfang.
UNLOAD	Zur ckspulen des Bandes und entladen.
LEAVE	Nicht zur ckspulen.
Standard:	REWIND
G ltig f r:	Komprimierung, Dekomprimierung

Hinweis: Wird zur Zeit ignoriert.

COMMENT

COMM

Angabe eines Kommentars.

Wird bei der Komprimierung in der Komprimatsdatei im Userheader (siehe FLMPUH, Kap. 3.3.21) gespeichert . Bei der Dekomprimierung werden davon die ersten 54 Zeichen im Protokoll angezeigt (FLM0487).

M gliche Werte:

1 - 256 Zeichen in der Form A'...', C'...', X'...' oder als String

Bei A'...' werden die Zeichen gem der internen FLAMtabelle E/A (siehe Anhang) in ASCII umkodiert.

Standard: kein Kommentar

G ltig f r: Komprimierung

Hinweis: Dieser Parameter darf bei Einsatz des KMEXITs nicht verwendet werden.

CRYPTOKEY

CRYPTOK

Schl ssel zur Ver- bzw. Entschl sselung des Komprimats

Mit der Angabe des Schl ssels wird bei der Komprimierung das eingestellte (siehe Parameter CRYPTOMODE) oder das bei der Dekomprimierung erkannte Verschl sselungsverfahren aktiviert.

M gliche Werte

1 - 64 Zeichen in der Form A'...', C'...', X'...' oder als String

Bei A'...' werden die Zeichen gem der internen FLAMtabelle E/A (siehe Anhang) in ASCII umkodiert.

Standard: kein Schl ssel

G ltig f r: Komprimierung, Dekomprimierung

Hinweis: Bitte beachten Sie, dass abdruckbare Zeichen nationaler Zeichens tze auch bei der Dekomprimierung identisch (d.h. bin r-kompatibel) eingegeben werden m ssen. F r heterogenen Austausch empfehlen wir die hexadezimale Eingabe X'...'.

CRYPTOMODE

CRYPTOM

Art des Verschl sselungsverfahrens

M gliche Werte:

AES	Advanced Encryption Standard
FLAM	das interne FLAM Verfahren

Standard: FLAM

G ltig f r: Komprimierung.

Hinweis: AES wurde mit FLAM V4.0 eingef hrt und ist in lteren Versionen nicht entschl sselbar.

Die Verschl sselung wird erst durch Angabe eines Schl ssels (Parameter CRYPTOKEY) aktiviert. Das Verschl sselungsverfahren ist im Komprimat gespeichert und muss zur Dekomprimierung nicht angegeben werden.

Verschl sselung setzt MODE=ADC oder NDC voraus. Ohne Angabe des Kompressionsmodus wird ADC eingestellt.

DECOMPRESS

D

Dekomprimierung.

[UNCOMPRESS]

Keine Werte

G ltig f r: Dekomprimierung

DEVICE

DEV

Ger tezuordnung f r die Komprimatsdatei.

M gliche Werte:

DISK	Plattenstation
TAPE	Bandstation
FLOPPY	Diskettenstation
STREAMER	Streamertape
USER	Benutzerspezifische Ein-/Ausgabe

Standard: DISK

G ltig f r: Komprimierung, Dekomprimierung

Hinweis: FLOPPY und STREAMER werden von VSE nicht unterst tzt. Bandstationen und Plattenstationen werden unter VSE aus den JCL-Anweisungen // DLBL und // TLBL erkannt.
Wenn die Benutzerschnittstelle Ein-/Ausgabe aktiviert werden soll, muss DEVICE=USER angegeben werden (siehe: Benutzer Ein-/Ausgabeschnittstelle).

DSORG

DS

Dateiorganisation f r die Komprimatsdatei.

[FCBTYPE]

M gliche Werte:

SAM	sequentiell	DTFSD / DTFMT / SAM-ESDS
ESDS	VSAM-ESDS	

KSDS	VSAM-KSDS
RRDS	VSAM-RRDS
Standard:	SAM
G ltig f r:	Komprimierung,Dekomprimierung

Hinweis: F r vorhanden VSAM-Dateien wird die Dateiorganisation dem Katalog entnommen.
Bei LIBR und POWER MEMBER wird dieser Parameter von FLAM auf sequentiell (SAM) eingestellt (Syntax siehe Dateinamen 3.1.4).

EXD10

EXD1

Benutzerausgang zur Bearbeitung der dekomprimierten Daten aktivieren.

M gliche Werte:

name	Name des Moduls (max. 8 Zeichen)
Standard:	kein Benutzerausgang
G ltig f r:	Dekomprimierung

Der Modul wird dynamisch geladen.

EXD20

EXD2

Benutzerausgang zur Bearbeitung des Komprimats aktivieren.

M gliche Werte:

name	Name des Moduls (max. 8 Zeichen)
Standard:	kein Benutzerausgang
G ltig f r:	Dekomprimierung

Der Modul wird dynamisch geladen.

EXK10

EXK1

Benutzerausgang zur Bearbeitung der Originaldaten aktivieren.

M gliche Werte:

name	Name des Moduls (max. 8 Zeichen)
Standard:	kein Benutzerausgang
G ltig f r:	Komprimierung

Der Modul wird dynamisch geladen.

EXK20

EXK2

Benutzerausgang zur Bearbeitung des Komprimats aktivieren.

M gliche Werte:

name	Name des Moduls (max. 8 Zeichen)
Standard:	kein Benutzerausgang
G ltig f r:	Komprimierung

Der Modul wird dynamisch geladen.

FILEINFO

FI

Dateinamen des Originals in Fileheader bernehmen.

M gliche Werte:

YES	Dateinamen in / aus FLAM-Fileheader bernehmen.
NO	Dateinamen nicht bernehmen (bei Komprimierung). Bei der Dekomprimierung wird ein Dateiname erzeugt (FILE0001-FILE9999), der f r Umsetzregeln verwendet werden kann.
Standard:	YES
G ltig f r:	Komprimierung

FLAMCODE

FLAMC

Code der FLAM-Syntax.

M gliche Werte:

EBCDIC	FLAM-Syntax wird in EBCDIC-Code erzeugt
ASCII	FLAM-Syntax wird in ASCII-Code erzeugt
Standard:	EBCDIC
G ltig f r:	Komprimierung

Hinweis: Liegen die Originaldaten im ASCII-Zeichensatz vor, werden mit FLAMCODE=ASCII h here Komprimierungswerte erreicht.

FLAMDDN

FLAMD

Symbolischer Dateiname f r die Komprimatsdatei.

[FLAMLINK]

M gliche Werte:

DD-NAME bis max. 8 Zeichen

> DD-Name bis max. 7 Zeichen (Dekomprimierung)

Standard: FLAMFIL

G ltig f r: Komprimierung, Dekomprimierung

Hinweis: Damit kann der DD-NAME f r den Zugriff auf die TLBL/DLBL-Anweisung ge ndert werden. Bei der Dekomprimierung bedeutet '>' vor dem DD-Namen, dass die Datei eine Liste von Komprimatsdateinamen enth lt.

FLAMFILE

FL

Dateiname f r die Komprimatsdatei.

M gliche Werte:

Dateiname bis max. 54 Zeichen

> Dateiname bis max. 53 Zeichen

Standard: kein Name

G ltig f r: Komprimierung, Dekomprimierung

Hinweis: Die Angabe eines Dateinamens ist nur f r VSAM-Dateien LIBR-MEMBER und POWER-MEMBER m glich. F r VSAM-Dateien ist die Angabe des Dateinamens alternativ zur Zuordnung der Datei ber DLBL-Anweisung.
Bei DTFSD/DTFMT-Dateien (nicht SAM-ESDS) kann die Zuweisung nicht ber den Dateinamen sondern nur ber die JCL-Anweisungen // DLBL bzw. // TLBL erfolgen.

Ein '>' vor dem Dateinamen hei t, die Datei enth lt eine Liste von zu dekomprimierenden Dateien.

FLAMIN

FLAMI

Dateiname f r die Eingabedatei.

M gliche Werte:

Dateiname bis max. 54 Zeichen

> Dateiname bis max. 53 Zeichen

Standard: kein Name

G ltig f r: Komprimierung

Hinweis: siehe Parameter FLAMFILE.

Ein '>' vor dem Dateinamen hei t, die Datei enth lt eine Liste von zu komprimierenden Dateien.

FLAMOUT

FLAMO

Dateiname f r die Ausgabedatei.

M gliche Werte:

Dateiname bis max. 54 Zeichen

Standard: kein Name

G ltig f r: Dekomprimierung

Hinweis: siehe Parameter FLAMFILE.

HEADER

HE

Fileheader erzeugen.

M gliche Werte:

YES Fileheader erzeugen

NO kein Fileheader erzeugen

Standard: YES

G ltig f r: Komprimierung

Hinweis: Der Header besteht aus drei Teilen. Der erste Teil ist unabh ngig vom Betriebssystem und enth lt kompatible Dateiattribute. Der zweite Teil ist betriebssystemabh ngig und enth lt spezielle Dateiattribute, die f r das jeweilige Betriebssystem spezifisch sind. Der dritte Teil ist optional und enth lt, durch den Parameter FILEINFO gesteuert, den Dateinamen.

FLAM bzw. FLAMUP werten den Fileheader aus, um die Datei m glichst mit den gleichen Eigenschaften wieder herzustellen. Das ist am einfachsten, wenn die Datei in der urspr nglichen Systemumgebung rekonstruiert werden soll, weil in diesem Fall auf den zweiten, betriebssystemspezifischen Teil des Headers zur ckgegriffen werden kann. In allen anderen F llen kann nur der erste Teil ausgewertet werden und die systemneutralen Attribute auf die systemspezifischen abgebildet werden.

HELP

Hilfe, Parameter ausgeben.

[?]

Keine Werte.

G ltig f r: Komprimierung, Dekomprimierung

Hinweis: Wenn die Hilfe-Funktion in der ersten Eingabezeile angefordert wird, werden die generierten

FLAM-Parameter mit ihren Werten ausgegeben und das Programm danach beendet.

INFO

I

Steuerung der Protokollierung.

M gliche Werte

YES	Meldungen und Statistik erzeugen und ausgeben.
NO	keine Meldungen ausgeben
HOLD	Die Parameter zur Komprimierung oder Dekomprimierung ausgeben, aber die Komprimierung bzw. Dekomprimierung nicht durchf hren
Standard:	YES
G ltig f r:	Komprimierung, Dekomprimierung

Hinweis: Der INFO-Parameter sollte in der ersten Eingabezeile stehen, da er sonst f r die Protokollierung der Parametereingaben ohne Wirkung ist. Die Statistik gibt Auskunft ber ben tigte Programmlaufzeit und Rechenzeit. Au erdem werden Satz- und Byteanzahl der Eingabe und Ausgabe ermittelt. Beim Dekomprimieren von relativen Dateien wird zus tzlich noch die um die L cken verminderte Satzanzahl ausgewiesen. Beim Konvertieren in ein fixes Format wird die gegebenenfalls ge nderte Byteanzahl ausgegeben.

INFO ist durch den erweiterten Parameter SHOW abgel st worden.

IDDN

Symbolischer Dateiname f r die Eingabedatei.

[ILINK]

M gliche Werte:

DD-NAME bis max. 8 Zeichen

> DD-Name bis max. 7 Zeichen

Standard:	FLAMIN
G ltig f r:	Komprimierung

Hinweis: Damit kann der DD-NAME f r den Zugriff auf die TLBL/DLBL-Anweisung ge ndert werden.

Ein '>' vor dem Dateinamen hei t, die Datei enth lt eine Liste von zu komprimierenden Dateien.

IBLKSIZE

IBLK

Logische Blockl nge f r die Eingabedatei.

M gliche Werte:

0 bis 32760

Standard: 32760 Byte

G ltig f r: Komprimierung

Hinweis: Dieser Parameter nur f r DTFSD/DTFMT Dateien notwendig.

CLOSDISP

ICLO

Endeverarbeitung f r Eingabedatei auf Band.

M gliche Werte:

REWIND	Zur ckspulen des Bandes an den Anfang
UNLOAD	Zur ckspulen des Bandes und entladen
LEAVE	Nicht zur ckspulen
Standard:	REWIND
G ltig f r:	Komprimierung

Hinweis: Wird zur Zeit ignoriert.

IDEVICE

IDEV

Ger tezuordnung f r die Eingabedatei.

M gliche Werte:

DISK	Plattenstation
TAPE	Bandstation
FLOPPY	Diskettenstation
STREAMER	Streamertape
USER	Benutzerspezifische Ein-/Ausgabe
Standard:	DISK
G ltig f r:	Komprimierung

Hinweis: siehe Parameter DEVICE.
Wenn die Benutzerschnittstelle Ein-/Ausgabe aktiviert werden soll, muss IDEVICE=USER angegeben werden (siehe: Benutzer Ein-/Ausgabeschnittstelle).

IDSORG

Dateiorganisation f r die Eingabedatei.

[IFCBTYPE]

M gliche Werte:

SAM	sequentiell	DTFSD / DTFMT / SAM-ESDS
ESDS	VSAM-ESDS	
KSDS	VSAM-KSDS	
RRDS	VSAM-RRDS	

Standard: SAM

G ltig f r: Komprimierung

Hinweis: siehe Parameter DSORG.

IKEYLEN

IKEYL

Schl ssell nge der Eingabedatei.

M gliche Werte:

0, 1 - 255

Standard: 0 = Kein Schl ssel

G ltig f r: Komprimierung

Hinweis: Bei KSDS Dateien wird die Schl ssell nge dem Katalog entnommen.

IKEYPOS

IKEY

Schl sselposition der Eingabedatei.

M gliche Werte:

0, 1 bis Satzl nge minus Schl ssell nge

Standard: 1 wenn Schl ssel vorhanden; sonst 0

G ltig f r: Komprimierung

Hinweis: Bei KSDS Dateien wird die Schl sselposition dem Katalog entnommen.

IRECSIZE

IRECS

Satzl nge der Eingabedatei (netto, ohne Satzl ngenfelder).

[ILRECL]
M gliche Werte:

0 bis 32760

Standard: 32750

G ltig f r: Komprimierung

Hinweis: Dieser Parameter nur f r DTFSD/DTFMT Dateien notwendig.

IRECFORM

IRECF

Satzformat f r die Eingabedatei.

[IRECFM]

M gliche Werte:

FIX	fixe Satzl nge
VAR	variable Satzl nge
UNDEF	Satzl nge undefiniert
FIXBLK	fix geblockt
VARBLK	variabel geblockt
SPNBLK	spanned geblockt
Standard:	VARBLK, variabel geblocktes Satzformat
G ltig f r:	Komprimierung

Hinweis: Dieser Parameter nur f r DTFSD/DTFMT Dateien notwendig.

IRECFM

Satzformat f r die Eingabedatei.

[IRECFORM]

M gliche Werte:

FIX	fixe Satzl nge
VAR	variable Satzl nge
UNDEF	satzl nge undefiniert
FB	fix geblockt
VB	variabel geblockt
VBS	spanned geblockt
Standard:	VB, variabel geblocktes Satzformat
G ltig f r:	Komprimierung

Hinweis: siehe Parameter IRECFORM.

IRECDEL

IREC

Satztrenner f r Eingabe-Originaldatei.

M gliche Werte:

String bis 4 Zeichen

Standard:	kein Satztrenner
G ltig f r:	Komprimierung

Hinweis: Wird von FLAM unter VSE nicht ausgewertet.

KEYDISP

KEYD

Schl sselbehandlung beim Dekomprimieren

OLD	Die S tze der Originaldatei werden wieder so erzeugt, wie sie eingelesen wurden. (Schl ssel + Daten)
DEL	Wenn die Originaldatei eine Schl ssell nge ungleich 0 aufweist, wird der Schl ssel entfernt.
NEW	Wenn die Ausgabedatei eine Schl ssell nge ungleich 0 aufweist, wird an der Schl sselposition in der Schl ssell nge eine fortlaufende Satznummer als abdruckbarer Schl ssel eingef gt.
Standard:	OLD
G ltig f r:	Dekomprimierung

Hinweis: Damit wird die automatische Konvertierung von importierten indexsequentiellen Dateien in sequentielle Dateien m glich.

KEYLEN

KEYL

Schl ssell nge einer indexsequentiellen Komprimatsdatei.

M gliche Werte:
0, 1 - 255

Standard:	0 (Kein Schl ssel)
G ltig f r:	Komprimierung, Dekomprimierung

Hinweis: F r eine vorhandene KSDS Datei wird die Schl ssell nge dem Katalog entnommen.

KMEXIT

KME

Anwendungsprogramm zur Schl sselverwaltung bei Ver-/Entschl sselung aktivieren.

M gliche Werte:

name	Name des Moduls (max. 8 Zeichen)
Standard:	kein Exit
G ltig f r:	Verschl sselung, Entschl sselung

Der Modul wird dynamisch geladen.

Hinweis: Hiermit kann ein Schl ssel zur Ver-/Entschl sselung bereitgestellt werden (siehe Kap. 3.5.5).
Dieser Schl ssel berschreibt eine evtl. CRYPTOKEY-Angabe.

KMPARM Parameter f r den KMEXIT

KMP Diese Parameter werden an das Anwendungsprogramm zur Schl sselverwaltung bergeben (siehe Kapitel 3.5.5 KMEXIT).

M gliche Werte:

1 - 256 Zeichen in der Form A’...’, C’...’, X’...’ oder als String

Bei A’...’ werden die Zeichen gem der internen FLAMtabelle E/A (siehe Anhang) in ASCII umkodiert.

Standard: keine Parameter

G ltig f r: Verschl sselung, Entschl sselung

Hinweis: Dieser Parameter berschreibt eine evtl. COMMENT-Angabe.

MAXBUFFER Maximale Gr e der Matrix.

MAXB Entweder Angabe eines Wertes zwischen 0 und 7

Wert:	0	1	2	3	4	5	6	7
entspricht Kbyte:	32	32	64	128	256	512	1024	2048

oder Angabe der Matrixgr e in KBytes.

Minimaler Wert: 8; maximaler Wert 2047

oder Angabe der Matrixgr e in Bytes.

Minimaler Wert: 2048

Standard: 64 KByte

G ltig f r: Komprimierung im MODE CX8/VR8

Hinweis: F r MODE=ADC wird dieser Parameter ignoriert.

Im VSE werden zur Beschleunigung Doppelpuffer angelegt, d.h. der Speicherbedarf ist doppelt so gro   wie angegeben.

Die Information der Puffergr e ist im Komprimat gespeichert und muss bei der Dekomprimierung nicht angegeben werden.

MAXRECORDS

MAXR

Maximale Anzahl von S tzen, die zusammen in einer Matrix komprimiert werden.

M gliche Werte:

1 - 255	f r MODE=CX7, CX8, VR8
1 - 4095	f r MODE=ADC
Standard:	255, 4095
G ltig f r:	Komprimierung

Hinweis: Bei CX7, CX8, VR8 werden gr   ere Werte auf das erlaubte Maximum reduziert.

MAXSIZE

MAXS

Maximale Satzl nge f r die Komprimatsdatei.

(Netto, ohne Satzl ngenfelder)

M gliche Werte:

80 - 32760

Standard:	512 Bytes
G ltig f r:	Komprimierung

Hinweis: Die Satzl nge der Komprimatsdatei ist unabh ngig von der Satzl nge der Originaldatei. Dieser Parameter sollte deshalb ausschlie lich aus Gesichtspunkten der Effizienz und Funktionalit t gew hlt werden. Um keinen Verschnitt im Komprimat zu erzeugen, sollte bei fixem Satzformat die Blockgr e ein ganzes Vielfaches der Satzl nge sein.

Durch die Erfordernisse eines Filetransfers k nnen andere Satzl ngen optimal oder notwendig sein (z.B: 80 Bytes fix f r RJE von IBM oder 2036 Bytes fix f r den Austausch zwischen SINIX und BS2000 mit FT-BS2000).

MODE

MO

Komprimierungsvariante.

M gliche Werte:

ADC	8-Bit Komprimat h chster Effizienz
CX7	transformierbares 7-Bit Komprimat
CX8	8-Bit Komprimat (Laufzeit optimiert)
VR8	8-Bit Komprimat (Speicherplatz optimiert)
NDC	keine Komprimierung
Standard:	VR8
G ltig f r:	Komprimierung

Hinweis: Der Modus der Komprimierung ist besonders bei Daten bertragung von Bedeutung. Lokal sollten nur die 8-Bit Codierungen des Komprimats (CX8/VR8/ADC) benutzt werden (h here Effizienz).

Bei   bertragung auf transparenten Leitungen ist ebenfalls der Modus (CX8/VR8/ADC) zu benutzen. Bei der   bertragung von komprimierten Textdaten (nur druckbare Zeichen, keine Steuerzeichen und Tabulatorzeichen)   ber nicht transparente Leitungen, kann die 7-Bit Codierung (CX7) verwendet werden. . Eine im CX7-Modus erzeugte FLAMFILE kann und darf w hrend eines Filetransfers im Zeichensatz umcodiert werden. Bei der Dekomprimierung werden die Daten dann in diesem Zeichensatz erstellt.
Da der CX7-Mode keine sicheren bin ren Checksummen enthalten kann, muss das verwendete File Transfer Programm f r Integrit t und Vollst ndigkeit sorgen. Ansonsten k nnten verf lschte Daten zu einem Fehler bei der Dekomprimierung f hren.

NDC (keine Kompression) ist sinnvoll bei Daten, die nicht (oder nur unwesentlich) komprimiert werden k nnen, z.B. bei erneuter Kompression von FLAMFILEs. Die Daten werden aber gem   der FLAM-Syntax f r ADC-Komprimate verpackt, verschleiert, gesichert und ggf. zus tzlich verschl sselt.

MO=ADC/NDC ist erforderlich f r CRYPTOMODE=AES oder SECUREINFO=YES.

Die Information ist im Komprimat gespeichert und muss zur Dekomprimierung nicht angegeben werden.

MSGDDN
MSGD

Symbolischer Dateiname f r die Meldungsausgabedatei.

[MSGLINK]

M gliche Werte:

DD-NAME bis max. 8 Zeichen

Standard: FLAMMSG

G ltig f r: Komprimierung, Dekomprimierung

Hinweis: Damit kann der DD-NAME f r die DLBL-Anweisung ge ndert werden.

MSGDISP

MSGD

Ger teauswahl f r die Meldungsausgabe.

M gliche Werte:

TERMINAL	wird nicht unterst tzt
MSGFILE	Ausgabe in die Listdatei VSAM-ESDS
SYSTEM	Ausgabe auf dem Drucker $$LST
Standard:	SYSTEM
G ltig f r:	Komprimierung, Dekomprimierung

Hinweis: Der MSGDISP-Parameter sollte in der ersten Eingabezeile stehen, da er sonst keine Wirkung hat.

MSGFILE

MSGF

Dateiname f r die Meldungsausgabedatei.

M gliche Werte:

Dateiname bis max. 54 Zeichen

Standard:	kein Name
G ltig f r:	Komprimierung, Dekomprimierung

Hinweis: Die Angabe des Dateinamens ist alternativ zur Zuordnung der Datei ber ein DLBL-Anweisung.
FLAM verwendet dabei eine Satzl nge von 133-Bytes.

OBLKSIZE

OBLK

Blockl nge f r die Ausgabedatei.

M gliche Werte:	0 bis 32760
Standard:	32760 Bytes bzw. der Wert aus Fileheader
G ltig f r:	Dekomprimierung

Hinweis: Dieser Parameter nur f r DTFSD/DTFMT Dateien notwendig. Bei neu zu erstellenden VSAM-Dateien dient dieser Parameter zur Errechnung des CI-SIZE.

OCLOSDISP

OCLO

Endeverarbeitung f r Ausgabedatei auf Band.

M gliche Werte:

REWIND	Zur ckspulen des Bandes an den Anfang
UNLOAD	Zur ckspulen des Bandes und entladen
LEAVE	Nicht zur ckspulen
Standard:	REWIND
G ltig f r:	Dekomprimierung

Hinweis: Wird zur Zeit nicht unterst tzt.

ODDN

Symbolischer Dateiname f r die Ausgabedatei.

[OLINK]

M gliche Werte:

DD-NAME bis max. 8 Zeichen

Standard:	FLAMOUT
G ltig f r:	Dekomprimierung

Hinweis: Damit kann der DD-NAME f r den Zugriff auf die DLBL/TLBL-Anweisung ge ndert werden.

ODEVICE

ODEV

Ger tezuordnung f r die Ausgabedatei.

M gliche Werte:

DISK	Plattenstation
TAPE	Bandstation
FLOPPY	Diskettenstation
STREAMER	Streamertape
USER	Benutzer Ein-/Ausgabe
Standard:	DISK
G ltig f r:	Dekomprimierung

Hinweis: siehe Parameter DEVICE.

Wenn die Benutzerschnittstelle f r Ein-/Ausgabe aktiviert werden soll, muss ODEVICE=USER angegeben werden (siehe: Benutzer Ein-/Ausgabeschnittstelle).

ODSORG
ODSO

Dateiorganisation f r die Ausgabedatei.

[OFCBTYPE]

M gliche Werte:

SAM	sequentiell	DTFSD / DTFMT / SAM-ESDS
ESDS	VSAM-ESDS	
KSDS	VSAM-KSDS	
RRDS	VSAM-RRDS	

Standard: SAM

G ltig f r: Dekomprimierung

Hinweis: siehe Parameter DSORG.

OKEYLEN
OKEYL

Schl ssell nge der Ausgabedatei.

M gliche Werte:

0, 1 - 255

Standard: 8 bzw. der Wert aus Fileheader

G ltig f r: Dekomprimierung

Hinweis: F r eine vorhandene KSDS Datei wird die Schl ssell nge dem Katalog entnommen.

OKEYPOS
OKEYP

Schl sselposition der Ausgabedatei.

M gliche Werte:

0, 1 bis Satzl nge minus Schl ssell nge

Standard: 1 bzw. der Wert aus Fileheader

G ltig f r: Dekomprimierung

Hinweis: F r eine vorhandene KSDS Datei wird die Schl sselposition dem Katalog entnommen.

ORECSIZE
ORECS

Satzl nge f r die Ausgabedatei.

(Netto, ohne Satzl ngenfelder)

M gliche Werte: 0 bis 32760

Standard: 32752 Bytes oder Wert aus Fileheader

G ltig f r: Dekomprimierung

Hinweis: Dieser Wert ist nur anzugeben, wenn die Satzl nge gegen ber dem Original ver ndert werden soll. Bei VSAM-Dateien wird die maximale Satzl nge dem Katalog entnommen.

ORECFORM

ORECF

Satzformat f r die Ausgabedatei.

M gliche Werte:

FIX	fixe Satzl nge
VAR	variable Satzl nge
UNDEF	satzl nge undefiniert
FIXBLK	fix geblockt
VARBLK	variabel geblockt
SPNBLK	spanned geblockt
Standard:	VARBLK oder Wert aus Fileheader
G ltig f r:	Dekomprimierung

Hinweis: Dieser Wert ist nur anzugeben, wenn das Satzformat gegen ber dem Original ver ndert werden soll.

ORECFM

Satzformat f r die Ausgabedatei.

M gliche Werte:

FIX	fixe Satzl nge
VAR	variable Satzl nge
UNDEF	satzl nge undefiniert
FB	fix geblockt
VB	variabel geblockt
VBS	spanned geblockt
Standard:	VB, variabel geblocktes Satzformat
G ltig f r:	Dekomprimierung

Hinweis: Dieser Wert ist nur anzugeben, wenn das Satzformat gegen ber dem Original ver ndert werden soll.

ORECDEL

ORECD

Satztrenner f r Ausgabedatei.

M gliche Werte:

String bis 4 Zeichen

Standard: kein Satztrenner

G ltig f r: Dekomprimierung

Hinweis: Wird von FLAM unter VSE nicht ausgewertet.

PARDDN

Symbolischer Dateiname f r die Parameterdatei (nur VSAM).

M gliche Werte:

DD-NAME bis max. 8 Zeichen

Standard: FLAMPAR

G ltig f r: Komprimierung, Dekomprimierung

Hinweis: Damit kann der DD-NAME f r den Zugriff auf die DLBL-Anweisung ge ndert werden. Wenn kein symbolischer Dateiname f r die Parameterdatei vereinbart ist (PARDDN= (NONE)), wird kein Versuch gemacht, aus dieser Datei zu lesen. Wenn die Parameterdatei nicht vorhanden oder leer ist, wird kein Fehler gemeldet. Da f r die Parameterdatei keine Dateiattribute eingestellte werden k nnen, d rfen nur VSAM-Dateien verwendet werden.

PARFILE

PARF

Dateiname f r die Parameterdatei.

M gliche Werte:

Dateiname bis max. 54 Zeichen (nur VSAM-Dateien)

Standard: kein Name

G ltig f r: Komprimierung, Dekomprimierung

Hinweis: Die Angabe des Dateinamens ist alternativ zur Zuordnung der Datei ber die DLBL-Anweisung. Diese Datei wird nur ben tigt, wenn zus tzlich Parameter aus einer katalogisierten Datei gelesen werden sollen.

PASSWORD

PASSW

PASSWORD zur Ver- bzw. Entschl sselung des Komprimats

M gliche Werte

1 - 64 Zeichen in der Form C'...', X'...' oder als String

Standard: kein Passwort

G ltig f r: Komprimierung, Dekomprimierung

Hinweis: Bitte beachten Sie, dass abdruckbare Zeichen nationaler Zeichens tze auch bei der Dekomprimierung identisch (d.h. bin r-kompatibel) eingegeben werden m ssen. F r heterogenen Austausch empfehlen wir die hexadezimale Eingabe X'...'.

Dieser Parameter ist identisch zu CRYPTOKEY.

RECDEL

RECD

Satztrenner f r Komprimatsdatei.

M gliche Werte:

String bis 4 Zeichen

Standard:	kein Satztrenner
G ltig f r:	Komprimierung, Dekomprimierung

Hinweis: Wird von FLAM unter VSE nicht ausgewertet.

RECFORM

RECF

Satzformat f r die Komprimatsdatei.

M gliche Werte:

FIX	fixe Satzl nge
VAR	variable Satzl nge
UNDEF	Satzl nge undefiniert
FIXBLK	fix geblockt
VARBLK	variabel geblockt
SPNBLK	spanned geblockt
Standard:	FIXBLK
G ltig f r:	Komprimierung, Dekomprimierung

Hinweis: Das Satzformat f r die Komprimatsdatei ist unabh ngig von der Originaldatei. Es sollten vorzugsweise fixe S tze benutzt werden.

RECFM

Satzformat f r die Komprimatsdatei.

M gliche Werte:

FIX	fixe Satzl nge
VAR	variable Satzl nge
UNDEF	Satzl nge undefiniert
FB	fix geblockt
VB	variabel geblockt
VBS	spanned geblockt
Standard:	FB
G ltig f r:	Komprimierung

Hinweis: Das Satzformat f r die Komprimatsdatei ist unabh ngig von der Originaldatei. Es sollten vorzugsweise fixe S tze benutzt werden.

SHOW

SH

Steuerung der Protokollierung

M gliche Werte:

ALL	Alle Meldungen und die Statistik erzeugen und ausgeben
NONE	Keine Meldungen ausgeben
ATTRIBUT	Die Parameter zur Komprimierung oder Dekomprimierung ausgeben, aber die Verarbeitung nicht durchf hren
ERROR	Nur Fehlermeldungen und Programm-endemeldung ausgeben
DIR	Die Namen aller Dateien mit Eigenschaften werden aufgelistet, die verarbeitet werden sollen.
Standard:	ALL
G ltig f r:	Komprimierung, Dekomprimierung

Hinweis: Der SHOW-Parameter sollte in der ersten Eingabezeile stehen, da er sonst f r die Protokollierung der Parametereingaben ohne Wirkung ist. Die Statistik gibt Auskunft ber ben tigte Programmlaufzeit und Rechenzeit. Au erdem werden Satz- und Byteanzahl der Eingabe und Ausgabe ermittelt. Beim Dekomprimieren von relativen Dateien wird zus tzlich noch die um die L cken verminderte Satzanzahl ausgewiesen. Beim Konvertieren in ein fixes Format wird die gegebenenfalls ge nderte Byteanzahl ausgegeben. Dieser Parameter entspricht dem INFO-Parameter (siehe: INFO).

TRANSLATE

TRA

Code-Konvertierung.

<CODE>

M gliche Werte:

E/A	konvertiert EBCDIC nach ASCII
A/E	konvertiert ASCII nach EBCDIC
TRA2E00	konvertiert ISO 8859-1 nach IBM 273
TRE2A00	konvertiert IBM 273 nach ISO8859-1

name — Name eines Datenmoduls (1-8 Zeichen), der eine 256 Byte lange bersetzungstabelle f r die Umcodierung enth lt

Standard: keine Code-Konvertierung

G ltig f r: Komprimierung, Dekomprimierung

Hinweis: Mit dieser Funktion k nnen die Originaldaten vor der Komprimierung bzw. vor dem Speichern zeichenweise bersetzt werden.

Bei Angabe eines Namens wird ein Tabellenmodul dynamisch geladen.

Codekonvertierungen k nnen bei Daten bertragungen zwischen unterschiedlichen Systemen erforderlich sein. Die Codekonvertierung kann in jedem System erfolgen, sollte aber auf dem Zielsystem durchgef hrt werden, da dort FLAM die f r das System geeigneten bersetzungstabellen enth lt.

Beispiel:

```
CODETAB

CSECT

TAB

DC       256AL1(*-TAB)

ORG      TAB+X'0C'

DC       X'F1'

ORG      TAB+C'A'

DC       C'B'

ORG

END
```

Bei Eingabe von TRA=CODETAB werden die Originaldaten konvertiert: Von X'0C' nach X'F1' und jeder Buchstabe A nach B.

TRUNCATE

TRU

Ausgabesatz verk rzen.

<OPTION=CUT/NOCUT>

M gliche Werte:

YES — Ist der dekomprimierte Satz l nger als in der Ausgabe zugewiesen, wird der Satz verk rzt

NO	l ngere S tze werden nicht gek rzt (kommen l ngere S tze vor, wird abgebrochen)
Standard:	NO
G ltig f r:	Dekomprimierung

3.1.2 JCL-Anweisungen f r FLAM

FLAM wird wie andere Utilities durch das EXEC-Kommando aufgerufen:

```
//  EXEC FLAM,...
```

Alle FLAM-Module ben tigen zum Ablauf eine Zuordnung zu den jeweils benutzten Datenbest nden.

Da zur Laufzeit verschiedene FLAM-Module nachgeladen werden, wird die Zuweisung einer Load Library mit diesen Modulen gebraucht (wenn nicht in Standardlabel):

```
// DLBL FLAMLIB,'vse.flam.lib',0,VSAM,CAT=VSESPUC

// LIBDEF PHASE,SEARCH=(FLAMLIB.TEST)
```

FLAM, FLAMUP bernehmen aus dem DLBL- bzw. TLBL-Statement der JCL-Kommandos bzw. den VSAM-Katalogeintr gen f r die Eingabe- und Komprimatsdatei eine Reihe von Attributen.

Statement f r die Eingabedatei beim Komprimieren:

a) SAM DTFSD :

```
//  ASSGN SYSnnn,cuu
//  DLBL  FLAMIN,'input.datei',0,SD,BLKSIZE=nnnn
//  EXTENT SYSnnn,volume
```

b) SAM DTFMT :

```
//  ASSGN SYSnnn,cuu
//  TLBL FLAMIN,'input.datei'  ;bei Standard LABEL Tape,
```

oder

```
//  TLBL FLAMIN                ; bei NO-LABEL Tape
```

c) VSAM-Dateien:

```
//  DLBL FLAMIN,'vsam.datei',0,VSAM,CAT=catddn
```

Unabh ngig von Parametern werden im Fileheader-Satz bernommen:

DSORG	SAM	(LIBR- POWER-MEMBER)
	SAM	(DTFSD, DTFMT)
	SAM	(SAM-ESDS abh ngig vom Dateinamen bzw. vom DLBL-Statement)
	ESDS, RRDS, KSDS (VSAM)	

RECFORM FIX, VAR,FIXBLK, VARBLK, SPNBLK, UNDEF (je nach DSORG)

RECSIZE wert

BLKSIZE wert (f r VSAM: Control Interval Size)

Bei index-sequentiellen Dateien werden zus tzlich folgende Attribute ber cksichtigt:

RKP wert

KEYLEN wert

Je nach Parameter FILEINFO wird auch der Dateiname der Eingabe bernommen.

Alle oben aufgef hrten Dateieigenschaften werden beim Dekomprimieren in VSE/ESA wiederhergestellt, sofern keine Dateiumsetzung gew nscht wird. In anderen Betriebssystemen werden diese Attribute (sofern m glich) auf quivalente Attribute des jeweiligen Systems abgebildet. VSAM-Dateien k nnen aber nur mit einen Datenbereich bzw. einem Daten- und einem INDEX-Bereich angelegt werden. Im anderen Fall muss der Anwender die Datei mit IDCAMS anlegen.

Statement f r die Ausgabedatei beim Dekomprimieren:

a) SAM DTFSD :

```
// ASSGN SYSnnn,cuu
// DLBL FLAMOUT,'output.datei',0,SD,BLKSIZE=nnnn
// EXTENT SYSnnn,volume,,0,blkcyl,size
```

b) SAM DTFMT :

```
// ASSGN SYSnnn,cuu
// TLBL FLAMOUT,'output.datei'      ;bei Standart LABEL
                                    Tape, oder

// TLBL  FLAMOUT                    ;bei NO-LABEL Tape
```

c) VSAM-Dateien:

```
// DLBL  FLAMOUT,'vsam.datei',0,VSAM,CAT=catddn
```

Statements f r die Komprimatsdatei:

a) SAM DTFSD :

```
// ASSGN SYSnnn,cuu
// DLBL  FLAMFIL,'flamfile',0,SD,BLKSIZE=nnnn
```



```
// EXTENT SYSnnn,volume,,0,blkcyl,size
```

b) SAM DTFMT :

```
// ASSGN SYSnnn,cuu
// TLBL FLAMFIL,'flamfile'   ;bei STANDARD-LABEL
                             Tape, oder

// TLBL FLAMFIL              ;bei NO-LABEL Tape
```

c) VSAM-Dateien:

```
// DLBL FLAMFIL,'vsam.flamfile',0,VSAM,CAT=catddn
```

Folgende Datei-Attribute sind zul ssig:

DSORG	SAM (LIBR- POWER-MEMBER) SAM (DTFSD, DTFMT) SAM (SAM-ESDS abh ngig vom Dateinamen bzw. vom DLBL-Statement) ESDS, RRDS, KSDS (VSAM)
RECFORM	FIX, VAR,FIXBLK, VARBLK, SPNBLK, UNDEF (je nach DSORG)
RECSIZE	80 - 32760
BLKSIZE	wert

Hinweise:

Die Komprimatsdatei ist in der Regel etwa 60 bis 80 % kleiner als die Eingabedatei. Bei gro en Datenmengen sollte f r die Komprimatsdatei entsprechend Speicherplatz mit der JCL-Anweisung EXTENT f r SAM oder in der DEFINE CLUSTER Anweisung bei VSAM reserviert werden.
Die Laufzeit kann au erdem durch die Blockung g nstig beeinflusst werden, da durch eine gro e BLKSIZE die Anzahl der Ausgabeoperationen verringert werden kann.

Weiterhin ist zu beachten, dass durch die Wahl der Satzl nge wenig oder kein Verschnitt erzeugt wird. Im VSE/ESA sind deshalb feste Satzl ngen mit entsprechender Blockung zu bevorzugen.

Die **Protokollierung** erfolgt in eine JCL-Liste ber

```
$$ LST
```

durch POWER oder als Ausgabe in eine Datei

```
// DLBL FLAMMSG,'message.datei',0,VSAM,CAT=catddn
```


Die **Zuordnung der logischen Ger tenummer SYSnnn** erfolgt durch das FLAM FILE-IO-MODUL wie folgt:

Bei Angabe einer EXTENT-Anweisung zu der DLBL-Anweisung wird von FLAM die logische Ger tenummer der EXTENT-Anweisung verwendet. Ist in der EXTENT-Anweisung keine logische Ger tenummer angegeben worden, so wird SYS000 angenommen. Wenn f r SAM-Dateien keine EXTENT-Anweisung verwendet wurde werden die logischen Ger tenummern wie folgt zugeordnet:

Openmode	TLBL	DLBL	komp.	dekomp.
input	SYS008	SYS001	FLAMIN	FLAMFIL
output	SYS009	SYS002	FLAMFIL FLAMMSG	FLAMOUT FLAMMSG

Beispiele mit JCL siehe Kapitel 5.

3.1.3 Return Codes

Zur Ablaufsteuerung werden bei der Komprimierung durch FLAM folgende Return Codes ($RC) gesetzt:

$RC=0 Die Komprimierung war fehlerfrei

$RC=4 CLIMIT   berschritten.

$RC=6 Es konnten nicht alle Dateien komprimiert werden.

$RC=8 Fehler einfacher Art (wie falsche Parameter) wurden erkannt.

Hinweis: Anhand der Zusatzmeldungen kann der Anwender den Fehler bereinigen.

$RC=12 In der Regel liegen DMS Zugriffsfehler vor.

Hinweis: Anhand der Zusatzmeldungen kann der Anwender den Fehler bereinigen.

$RC=16 Fehler beim Zugriff auf die FLAMFILE oder Komprimierungsfehler.

Hinweis: Kann der Anwender den Fehler nicht beheben, so verst ndigen Sie bitte Ihren Vertriebspartner.

Zur Ablaufsteuerung werden bei der Dekomprimierung durch FLAM folgende Return Codes gesetzt:

$RC=0 Die Dekomprimierung war fehlerfrei

$RC=6 Es konnten nicht alle Dateien dekomprimiert werden.

$RC=8 Fehler einfacher Art (wie falsche Parameter) wurden erkannt.

Hinweis: Anhand der Zusatzmeldungen kann der Anwender den Fehler bereinigen.

$RC=12 In der Regel liegen DMS Zugriffsfehler vor.

Hinweis: Anhand der Zusatzmeldungen kann der Anwender den Fehler bereinigen.

$RC=16 Fehler beim Zugriff auf die FLAMFILE oder Dekomprimierungsfehler

Hinweis: Kann der Anwender den Fehler nicht beheben, so verst ndigen Sie bitte Ihren Vertriebspartner.

$RC=88 Die zugewiesene Datei ist keine FLAMFILE.

Hinweis: Kann zur Identifikation von nicht komprimierten Data Sets in Batch-Jobs genutzt werden.

Nur bei Returncode 0 ist eine Dekomprimierung ordnungsgem   erfolgt. Genauere Hinweise zu Fehlern sind den Fehlermeldungen FLM04xx in Kapitel 8 zu entnehmen.

3.1.4 Dateinamen

Alternative zur Zuweisung von Dateien ber JCL (DLBL) k nnen auch bestimmte Dateien ber Dateinamen zugewiesen werden.
Die Zuweisung von Dateien in den Parametern **FLAMIN**, **FLAMFILE**, **FLAMOUT**, **MSGFILE** und **PARFILE** ist nur f r VSAM-Dateien (SAM-ESDS, ESDS, RRDS, KSDS) und LIBR-MEMBER bzw. POWER-MEMBER erlaubt.
Die Namen folgen einer bestimmten Syntax anhand deren die Zugriffsart erkannt wird.

Syntax f r VSAM-Dateien:

catddn:*clustername*

catddn = Katalog DDNAME (1-7 Zeich.)
clustername = Dateiname im Katalog

Beispiel: FLAMIN=VSESPUC:FLAM.INPUT.DATEN

Syntax f r LIBR-MEMBER:

libddn.sublib(member.type)

libddn = VSE-LIBR DDNAME (1-7 Zeich.)
sublib = Sublibname (1-8 Zeich.)
member = Membername (1-8 Zeich.)
type = Membertyp (1-8 Zeich.)

Beispiel: FLAMV30.LIB(FLAMPAR.OBJ)

Es k nnen Member aus LIBR-Bibliotheken gelesen und geschrieben werden. FLAM unterst tzt beim Schreiben das RECORD-FORMAT mit FIX 80-BYTES und das BYTESTRING-FORMAT mit Recform=VAR oder UNDEF mit Satzl ngen bis 32760. Beim lesen im BYTESTRING-FORMAT muss der Anwender eine Satzl nge vorgeben andernfalls wird in Segmenten von 1024-Bytes gelesen, wobei das letzte Segment k rzer sein kann. Member mit Type OBJ oder PROC k nnen nur FIX 80-Bytes gelesen und geschrieben werden.

PHASEN werden von FLAM nicht unterst tzt.

Library und Sublibrary m ssen vorhanden und in den Standardlabels eingetragen sein.

Syntax f r POWER-MEMBER:

queue(jobname,jobnr,class,dispo,user,passw)

queue = Powerqueue (#LST,#PUN, #RDR)
jobname = Jobname (1-8 Zeich.)
jobnr = Jobnummer (1-65535)
class = Job Class (1 Zeich.)
dispo = Job disposition (1 Zeich.)

user	=	User-ID (1-8 Zeich.)
passw	=	Passwort (1-8 Zeich.)

Beispiel: #LST(JFLAMLIZ,2222,A,D,FLAM)

Member ohne User-ID (Originator oder Target) k nnen nicht gelesen werden. Das Passwort muss nur bei Bedarf eingeben werden.

Zum Schreiben in die Power Queue wird eine vorgegebene Jobnummer ignoriert (die Jobnummer wird von Power vergeben). Beim Schreiben werden von FLAM folgende Defaultwerte gesetzt: Jobname=FLAMPWR, Class=F, Dispo=K und Priority=9.

Bei Member aus der READER- und PUNCH-Queue (#RDR, #PUN) liest FLAM nur die Daten, das Vorschubsteuerzeichen wird nicht mit bergeben. Beim Schreiben in die READER- / PUNCH-Queue wird das Spoolrecordformat auf ASA gesetzt und ein Vorschub von 1 (Commandcode=x'40') pro Satz von FLAM eingestellt.

F r die PUNCH-Queue d rfen die S tze maximal 80-Bytes und f r die READER-Queue maximal 128-Bytes lang sein.

Beim Lesen der Member aus der LIST-Queue wird bei den Spoolrecordformaten ASA (ASA Control Character) und MCC (Machine Control Character) der 1-Byte Commandcode vor den Datensatz gestellt. Bei allen anderen Spoolrecordformaten werden nur die Daten weitergegeben.
Member die mit ASA oder MCC Steuerzeichen beim Komprimieren gelesen wurden, werden beim Dekomprimieren (schreiben) in die LIST-Queue auch mit diesem Vorschubsteuerzeichen wieder erzeugt. Bei allen anderen Member werden nur die Daten in die LIST-Queue geschrieben, in das Commandcodebyte x'00' eingetragen und das Spoolrecordformat auf Escape Mode (x'08') gestellt.

3.1.4.1 Dateinamensliste

Durch Voranstellen des Zeichens '>' (gr er) im Dateinamen oder DD-Namen der FLAM-Parameter FLAMIN oder IDDN kann f r die Komprimierung anstatt einer einzelnen Datei eine Dateiliste vorgegeben werden. In der Dateiliste d rfen nur Dateien stehen, die ber Dateinamen zugeordnet werden d rfen (siehe 3.1.4 Dateinamen).

Analog kann f r die Dekomprimierung eine Dateiliste ber FLAMFILE oder FLAMDDN vergeben werden.

In dieser Dateiliste muss jeder Dateiname in einem separaten Satz enthalten sein, f hrende oder folgende Leerzeichen (X'40') werden ignoriert. Ein beliebiger Kommentar kann nach dem 1. Leerzeichen hinter dem Dateinamen eingef gt werden. Der Dateiname muss der Syntax der Dateiname aus Kapitel 3.1.4 entsprechen.

Leers tze oder S tze mit einem Stern '*' in der 1. Spalte werden als Kommentar angesehen.

3.1.4.2 Wildcard-Syntax

Die Wildcard-Syntax darf beim FLAMIN, FLAMFILE UND FLAMOUT Parameter verwendet werden, und zwar beim Komprimieren nur mit dem FLAMIN, FLAMFILE Parameter und beim Dekomprimieren nur mit dem FLAMOUT, FLAMFILE Parameter.

Als Wildcards werden verwendet:

*	(Stern)	beliebige Zeichenfolge
%	(Prozent)	ein beliebiges Zeichen

Beispiele:

```
FLAMIN=VSESPUC:FLAMV30.LIB(*.OBJ)
```

Es werden alle OBJ-Member der Sublib FLAMV30.LIB im Katalog VSESPUC komprimiert/verschl sselt.

```
FLAMIN=#LST(*,*,H,*,SYSA)
```

Alle Listen des Users SYSA der Klasse H werden als Eingabe zugewiesen.

```
FLAMOUT=<DE*.;*=VSESPUC:USER.DE*'>
```

Eine Auswahl- und Umsetzvorschrift (siehe Kapitel 3.1.4.4) zur Selektion bestimmter Komprimate und deren zur Erstellung zu verwendenden Dateinamen.

```
FLAMFILE=#PUN(FLAM%%,*,F,K,SYSA)
```

Alle Komprimate der Punch-Queue des Users SYSA mit Jobnamen FLAM beginnend und mit zwei beliebigen Zeichen folgend, der Klasse F und der Disposition K werden zum Dekomprimieren zugewiesen.

3.1.4.3 Auswahlvorschrift bei der Dekomprimierung

Bei der Dekomprimierung k nnen die VSAM-Dateien durch FLAM selbstt tig angelegt werden. Dazu bedarf es eines g ltigen Dateinamens im Fileheader der FLAMFILE (d.h. die Parameter HEADER und FILEINFO d rfen bei der Komprimierung nicht auf NO gesetzt sein).

Zus tzlich kann eine FLAMFILE mehrere Dateien enthalten (Sammeldatei). Durch Angabe einer Auswahlvorschrift lassen sich gezielt Dateien aus einer Sammel-FLAMFILE dekomprimieren.

Eine Auswahlvorschrift wird zur Unterscheidung von einem ’echten’ Dateinamen in spitze Klammern ’<>’ gesetzt.

```
FLAMOUT=<VSESPUC:DATEI.A1>
```

Damit wird aus der FLAMFILE die Datei VSESPUC:DATEI.A1 dekomprimiert (Anmerkung: ohne spitze Klammern w rde die gesamte FLAMFILE in die Datei VSESPUC:DATEI.A1 dekomprimiert !). Dieser Name muss in einem Fileheader der FLAMFILE enthalten sein.

Sollten in der FLAMFILE noch weitere Komprimatsdateien enthalten sein, so werden sie durch die eindeutige Auswahlvorschrift ignoriert.

Sollen mehrere Dateien aus einer Sammeldatei dekomprimiert werden, so kann eine Wildcard-Syntax vorgegeben werden. Die einfachste Angabe ist der Stern allein:

```
FLAMOUT=<*>
```

Damit werden alle Dateien bzw. Member aus der FLAMFILE dekomprimiert und mit ihrem originalen Dateinamen auf der Platte, VSE-Libray oder POWER-Queue erstellt.

Implizit wird eine Auswahlvorschrift durch einen Stern am Anfang und am Ende erg nzt, d.h.

```
<DAT*ABC > entspricht  <*DAT*ABC*>
```

Bei der Analyse der Dateinamen synchronisiert sich FLAM auf die angegebene Zeichenfolge (Fragmentauswahl).

Beispiel: Die FLAMFILE enthalte die Daten VSESPUC:U.DAT1.A1 und VSESPUC:U.DAT2.B1. Die Angabe

```
...D,FLAMOUT=<DAT1>,...
```

dekomprimert nur die Datei VSESPUC:U.DAT1.A1.

Da hier kein Stern im Namen angegeben ist, wird die Dekomprimierung nach dem ersten Treffer beendet.

3.1.4.4 Umsetzvorschrift

Die einfache Auswahl von Dateien und Member zur Dekomprimierung mittels einer Auswahlvorschrift ist aber gerade bei Komprimaten, die unter fremden Betriebssystemen erstellt wurden (heterogener Komprimatsaustausch), in der Regel nicht gegeben. Die Dateinamen entsprechen gew hnlich nicht den Regeln des VSE-Betriebssystems und k nnen somit nicht ohne ˜ nderung verwendet werden.

Dazu kann als Parameter f r die Ausgabedatei eine Umsetzvorschrift angegeben werden. Diese Zeichenfolge beschreibt, wie aus einem selektierten Dateinamen ein neuer Name gebildet werden soll. Gleichzeitig wird eine Selektion genau der Dateien vorgenommen, die der Vorschrift entsprechen (Auswahlvorschrift).

Eine Umsetzvorschrift ist eine Auswahlvorschrift, die durch ein Gleichheitszeichen '=' und eine zweite Zeichenfolge erg nzt wird. Sie ist zur Unterscheidung von einem "echten" Dateinamen in spitze Klammern '<' '>'zu setzen. Die Vorschrift besteht aus einer Zeichenfolge, die den Stern '*' als Ersatzzeichen f r eine beliebige Anzahl Zeichen oder das Prozentzeichen '%' als Ersatz f r genau ein Zeichen enthalten darf. Zus tzlich ist ein Auslassungszeichen (Apostroph ') definiert.

Jedem Stern '*' oder Prozentzeichen '%' der Auswahlvorschrift muss ein Stern oder Prozentzeichen oder jeweils ein Apostroph in der Umsetzvorschrift zugeordnet sein.

Der Stern bedeutet, dass die Zeichenfolge aus der Eingabe in die Ausgabe bernommen werden soll. Analog wird bei '%' genau das an dieser Stelle stehende (beliebige) Zeichen bernommen.

Das Apostroph bewirkt, dass die durch Stern oder Prozentzeichen in der Eingabe repr sentierten Zeichenfolge oder Zeichen nicht in die Ausgabe bernommen werden soll. Die brigen Zeichen aus der Eingabe werden in die entsprechenden Zeichen aus der Umsetzvorschrift bersetzt. Dabei kann die L nge der Zeichenfolge beliebig ver ndert werden.

<USER.*=USER2.*>

Hier werden alle Dateinamen, beginnend mit USER, in die neue Kennung USER2 bersetzt. Der brige Namensteil bleibt erhalten.

<USER.DAT%B.*=USER.DECDAT%C.*>

Alle Dateien der Kennung USER erhalten den Pr fix DEC vor dem alten Namen. Dabei werden nur die Dateien ber cksichtigt, deren zweiter Namensteil mit DAT beginnt dem ein beliebiges Zeichen folgt und mit B endet. Dieses B wird im Namen der Ausgabedatei in C umgesetzt.

Insbesondere ist auch die leere Zeichenfolge in der Umsetzvorschrift zugelassen, um Zeichen zu l schen, z.B.:

```
<USER*UP*=USER.CMP**>
```

alter Name: **USER.FLAMUP00**
neuer Name: **USER.CMPFLAM00**

Der Namensteil UP ist in der Ausgabe nicht erw hnt und wird somit weggelassen.

Eine Umsetzvorschrift wird implizit erg nzt, z.B.:

```
<ASM.=CMP.> entspricht <*ASM.*='CMP.*>
```

Dies kann besonders bei Umsetzung der Dateinamen von Fremdsystemen verwendet werden.

Beispiel:

Die auf DEC/VMS erstellte FLAMFILE enth lt die Dateinamen

```
DUA1:[ABC]DE0051.;7

DUA1:[ABC]DE0052.;4

DUA1:[ABC]DE0080.;2

DUA1:[ABC]DE0152.;4
```

Dies entspricht der Angabe von Dateiversionen des Benutzers ABC auf dem Plattenvolume DUA1 mit der Versionsnummer nach dem Semikolon.

Um diese Dateien in VSE erstellen zu k nnen, kann z.B. folgende Umsetzvorschrift angegeben werden:

```
FLAMOUT=<DE*.;*=VSESPUC:USER.DE*'>
```

Dadurch wird der Namensvorspann DUA1:[ABC] implizit gel scht, der Namensteil mit DE als Anfang bernommen und um die Kennung erg nzt, der restliche Namensbestandteil gel scht.

```
VSESPUC:USER.DE0051

VSESPUC:USER.DE0052

VSESPUC:USER.DE0080

VSESPUC:USER.DE0152
```

Bisher wurde die Umsetzvorschrift nur f r die Dekomprimierung aus einer (Sammel-) FLAMFILE beschrieben.

Sie gilt aber auch bei der Komprimierung bei der gleichzeitigen Erstellung von mehreren Komprimaten in unterschiedlichen Dateien. Die Umsetzvorschrift bezieht sich hierbei auf die FLAMFILEs, deren Namen aus den Dateiliste der Eingabedateien (FLAMIN) gebildet werden.

Beispiel: Umsetzung bei der Komprimierung

```
// EXEC FLAM,SIZE=AUTO
C,FLAMIN=FLAMV30.LIB(*.OBJ),MODE=ADC,MAXS=80,
RECFORM=FIX,FLAMFILE=<*(*.OBJ)=*(*.CMP)>,END
/*
```

Es werden ALLE OBJ-Member der sublib=FLAMV30.LIB werden als Komprimat mit membername.CMP in die gleiche Sublib geschrieben.

Beispiel: Umsetzung bei der Dekomprimierung.

```
// EXEC FLAM,SIZE=AUTO
D,FLAMF=FLAMV30.LIB(*.CMP),
FLAMOUT=<FLAMV30.LIB(*)=FLAMTST.LIB(*)>,END
/*
```

Aus der lib.sublib FLAMV30.LIB werden alle Komprimatsmember mit dem Membertyp=CMP gelesen. Bei der Ausgabe werden alle Member ausgew hlt die als Original aus der FLAMV30.LIB stammen und dann unter gleichem Membernamen und Membertyp in die FLAMTST.LIB dekomprimiert.

Hinweis: Wurde eine Sammeldatei mit HEADER=YES aber FILEINFO=NO erstellt, so ist f r die jeweilige Datei kein Dateiname gespeichert.

Die einzelnen Dateien k nnen dann zur Dekomprimierung ber den internen Dateinamen FILE0001 (f r die 1. Datei) bis FILE9999 (f r die 9999. Datei) angesprochen werden:

```
...D,FLAMOUT=<FILE0003=VSESPUC:U.DAT.DREI>,..
```

f r die dritte Datei in der Sammeldatei; oder auch

```
...D,FLAMOUT=<FILE*=VSESPUC:U.DAT*>,..
```

zur Dekomprimierung aller Dateien gem Umsetzregel.

Anmerkung: Als "letzte Rettungsm glichkeit" bei automatischer Erstellung der Dekomprimate mit "unm glichen" Dateinamen fremder Betriebssysteme kann der Parameter FILEINFO=NO bei der Dekomprimierung angegeben werden. Damit werden die gespeicherten Dateinamen ignoriert und die internen Namen FILE0001 bis FILE9999 generiert. Diese m ssen dann per Umsetzvorschrift in g ltige Dateinamen umgesetzt werden.

3.1.4.5 Interne Dateinamen

Wurde eine Sammeldatei mit HEADER=YES aber FILEINFO=NO erstellt, so ist f r die jeweilige Datei kein Dateiname gespeichert. FLAM erkennt einen Dateiwechsel, hat aber keinen Dateinamen zum Anlegen zur Verf gung.

Die einzelnen Dateien k nnen dann zur Dekomprimierung ber die internen (generierten) Dateinamen FILE0...001 (f r die 1. Datei) bis FILE9...999 (f r die 9...999. Datei) angesprochen werden:

```
...D,FLAMOUT=<FILE0003=USER.DAT.DREI>,..
```

Die dritte Datei in der Sammel FLAMFILE soll dekomprimiert werden, die Ausgabedatei erh lt den Namen USER.DAT.DREI.

Die Angabe FILE0003 ist identisch zu FILE3. Im letzteren Fall k nnten aber nur max. 9 Dateien selektiert werden! Die Anzahl der Ziffern bestimmt damit die maximale Anzahl selektierbarer Dateinamen.

Es d rfen maximal 12 Ziffern angegeben werde (d.h. es sind in einer Sammel FLAMFILE bis zu 999.999.999.999 Dateien so selektierbar).

Als "letzte Rettungsm glichkeit" bei automatischer Erstellung der Dekomprimate mit "unm glichen" Dateinamen fremder Betriebssysteme kann der Parameter FILEINFO=NO bei der Dekomprimierung angegeben werden. Damit werden die gespeicherten Dateinamen ignoriert und die internen Namen FILE0001 bis FILE9999 generiert. Diese m ssen dann per Umsetzvorschrift in g ltige Dateinamen umgesetzt werden:

```
...D,FI=NO,FLAMOUT=<FILE*=USER.DAT*>,..
```

zur Dekomprimierung bis zu 9.999 Dateien gem Umsetzregel. Hier werden genau 4 Ziffern generiert, um neue Dateinamen zu erm glichen. Eine variable L nge wie bei der Einzelselektion ist nicht m glich.

3.1.5 Dateien f r gesplittete FLAMFILEs

Beim Splitt der FLAMFILE entstehen mehrere Dateien, die Fragmente des Komprimats enthalten. Diese Fragmente k nnen nicht jedes f r sich allein dekomprimiert werden. Fragmente verschiedener Komprimierungen k nnen nicht gemischt werden, selbst wenn die gleichen Daten komprimiert worden sind.

Die Fragmente k nnen entweder ber JCL vorgegeben werden oder FLAM allokiert diese Dateien selbstt tig (wie die FLAMFILE als Einzeldatei).

Es gen gt die Angabe des 1. Fragments. Weitere Dateien werden selbstt tig gesucht.

Die Angabe ist f r Komprimierung und Dekomprimierung gleich.

3.1.5.1 Namensregeln beim Splitt

Damit FLAM selbstt tig Dateien f r den Splitt anlegen bzw. erkennen kann, m ssen Regeln bez glich von Datei- oder DD-Namen eingehalten werden.

Dazu muss entweder der DD-Name oder der Dateiname eine Ziffernfolge enthalten, die durch FLAM hochgez hlt werden kann. Diese Ziffer muss nicht bei Eins beginnen, der ‚Startwert' kann beliebig angenommen werden. Danach d rfen im Namen keine L cken entstanden sein. Die Ziffern werden von rechts beginnend im Namen gesucht. Die Anzahl Ziffern bestimmt dabei die maximal m gliche Dateianzahl. So k nnen z.B. bei FLAM1 nur maximal 9 Namen angesprochen werden, bei FLAM01 max. 99 oder bei FLAM5 nur 5.

Variable DD-Namen empfehlen sich bei Vorgabe von Dateien per JCL, deren Dateinamen keiner solchen Regelung gen gen:

```
// DLBL FLAM01,'DAT.WORK.ESDS',,VSAM,CAT=LIMES
// DLBL FLAM02,'ASD.DAT.W231.A',,VSAM,CAT=LIMES
// EXEC FLAM,SIZE=AUTO,PARM='D,FLAMDDN=FLAM01'
```

Ansonsten empfiehlt sich die Verwendung variabler Dateinamen:

```
// DLBL FLAMFIL,'DAT.F001.ADC',,VSAM,CAT=LIMES
// EXEC FLAM,SIZE=AUTO,PARM=DECO
```

oder einfach als FLAM-Parameter:

```
FLAMFILE=LIMES:DAT.F001.ADC
```


Hier wird auf die Dateien DAT.F001.ADC, DAT.F002.ADC, ...usw. zugegriffen, bis alle Fragmente der FLAMFILE verarbeitet worden sind.

3.1.5.2 Dateiattribute beim Splitt

Bei der Komprimierung m ssen alle Dateien die gleiche Satzl nge haben. Dateiformat oder Dateiorganisation d rfen dabei differieren.

Hat die erste Datei z.B. eine fixe Satzl nge von 512 Byte, so darf die zweite nicht eine von 1024 Byte haben. Wohl w re aber eine Datei variabler Satzl nge mit LRECL=1024 erlaubt.

Bei Dateien variabler Satzl nge muss ein FLAMFILE-Satz gem der MAXSIZE-Angabe in jede Datei geschrieben werden k nnen. Bei MAXS=512 ist also die minimale Satzl nge 512 bei VSAM, bzw. 516 bei Magnetb ndern mit variablem Satzformat. Hier w re LRECL=1024 oder RECSIZE=1000 f r Folgedateien erlaubt.

Es empfiehlt sich aber, f r alle Fragmente der FLAM-FILE stets die gleichen Dateiattribute zu w hlen.

FLAM ist bei der Dekomprimierung tolerant gegen ber ‚falschen' Dateiattributen.

Es geschieht sehr h ufig, dass nach einem Filetransfer von einem PC die urspr nglich in FLAM eingestellte Satzl nge nicht beibehalten worden ist. So kann es z.B. geschehen, dass eine auf dem PC mit 512 Byte Satzl nge erstellte Datei auf dem Host als Datei mit fixer Satzl nge von 80 Byte angekommen ist. Solange die S tze nur ‚umgebrochen' sind und nicht etwa abgeschnitten, kann FLAM diese Dateien dekomprimieren.

Diese Eigenschaft wurde beim Splitt so erweitert, dass jedes Fragment zur Dekomprimierung unterschiedlich sein darf, da es auf unterschiedlichen Wegen auf den Rechner gelangt sein k nnte (mit unterschiedlichen Transferprogrammen und Einstellungen).

So ist z.B. die erste Datei eine sequentielle Datei mit fixer Satzl nge von 80 Byte, die zweite eine VSAM-ESDS Datei mit Satzl nge 1024, die dritte ein POWER Queue-Member mit variabler Satzl nge von 32752 Byte, usw.

Bitte beachten Sie, dass diese Eigenschaft auch von selbst erstellten Exits oder USER-I/Os ber cksichtigt werden m ssen, sobald sie beim Splitt im Einsatz sind.

3.2 Unterprogrammschnittstelle FLAMUP

Im folgenden werden die Schnittstellen in ASSEMBLER beschrieben. Die Tabelle zeigt, wie die verschiedenen Datentypen in COBOL und FORTRAN definiert werden m ssen.

Mit FLAMUP kann eine Datei vollst ndig komprimiert oder eine Komprimatsdatei dekomprimiert werden. Analog zum Dienstprogramm k nnen Parameter bergeben werden. FLAMUP verwendet die gleichen Parameter wie das Dienstprogramm. Alle Parameter k nnen ber die Generierung fest voreingestellt werden.

Assembler	Cobol	Fortran	Bedeutung
F	PIC S9 (8) COMP SYNC	INTEGER*4	ausgerichtetes Ganzwort
H	PIC S9 (4) COMP SYNC	INTEGER*2	ausgerichtetes Halbwort
CL*n*	PIC X (*n*) USAGE DISPLAY	CHARACTER* *n*	*n* abdruckbare Zeichen
XL*n*	PIC X (*n*)	CHARACTER* *n*	*n* bin re Zeichen

Die Pfeile bezeichnen die Richtung des Datenflusses:

→ das Feld ist vom rufenden Programm zu versorgen

↔ das Feld wird vom gerufenen Programm gef llt

↔ sowohl rufendes als auch gerufenes Programm versorgen das Feld

Parameter:

1 ← FILEID F Kennung

2 ← RETCO F Returncode

Einige g ngige Fehlercodes (siehe auch Kapitel 8.4):

=	**0**	Kein Fehler
=	**1**	S tze verk rzt
=	**9**	Climit berschritten
=	**10**	Datei ist keine FLAMFILE
=	**11**	FLAMFILE Formatfehler
=	**12**	Satzl ngenfehler
=	**13**	Dateil ngenfehler

=	**14**	Checksummenfehler
=	**15**	Originalsatz ist gr er als 32764 Bytes
=	**16**	Originalsatz ist gr er als Matrix -4
=	**20**	Unzul ssiger OPENMODE
=	**21**	Unzul ssige Gr e des Matrixpuffers
=	**22**	Unzul ssiges Kompressionsverfahren
=	**23**	Unzul ssiger Code in FLAMFILE
=	**24**	Unzul ssiger MAXRECORDS-Parameter
=	**25**	Unzul ssige Satzl nge MAXSIZE
=	**30**	FLAMFILE ist leer
=	**31**	FLAMFILE nicht zugeordnet
=	**32**	Unzul ssiger OPENMODE
=	**33**	Ung ltiger Dateityp
=	**34**	Ung ltiges Satzformat
=	**35**	Ung ltige Satzl nge
=	**36**	Ung ltige Blockl nge
=	**37**	Unzul ssige Schl sselposition (ungleich 1)
=	**38**	Ung ltige Schl ssell nge
=	**39**	Ung ltiger Dateiname
=	**x'Exxxxxxx'**	FLAMFIO-Fehler f r Originaldatei Eingabe
=	**x'Axxxxxxx'**	FLAMFIO-Fehler f r Originaldatei Ausgabe
=	**x'Fxxxxxxx'**	FLAMFIO-Fehler f r Komprimatsdatei
=	**x'Cxxxxxxx'**	FLAMFIO-Fehler f r Parameterdatei
=	**x'Dxxxxxxx'**	FLAMFIO-Fehler f r Meldungsdatei
=	**x'xFxxxxxx'**	Fehler der Datenverwaltung (VSAM,LIBR,POWER)
=	**40**	Modul oder Tabelle kann nicht geladen werden
=	**41**	Modul kann nicht aufgerufen werden
=	**42**	Modul kann nicht entladen werden
=	**43 - 49**	Fehlerabruch durch Exit
=	**52**	zuviele oder unzul ssige Schl ssel
=	**57**	unzul ssige Teilkomprimatsl nge
=	**60**	Syntaxfehler im Komprimat
=	**61**	Zu viele Z hler erkannt
=	**62**	L ngenfehler im Komprimat
=	**65**	Konsistenzpunkt falsch
=	**66**	Konsistenzpunkt falsch
=	**67**	Konsistenzpunkt falsch
=	**68**	Satzl ngenfehler in Matrix
=	**69**	Satznummer = 0 bei Sortierung
=	**70**	Version stimmt nicht
=	**71**	Stop-Bit V0 nicht gefunden
=	**72**	Stop-Bit V8 nicht gefunden
=	**73**	L nge Komprimat falsch
=	**74**	Pr fzeichenfehler
=	**75**	Syntaxfehler im Komprimat

=	**77**		Konsistenzsatz zu kurz
=	**78**		Spaltenl nge unlogisch
=	**80**		Syntaxfehler bei Parametereingabe
=	**81**		Unbekannter Parameter (Schl sselwort)
=	**82**		Unbekannter Parameterwert
=	**83**		Parameterwert nicht dezimal
=	**84**		Parameterwert zu lang
=	**96**		Keinen Dateinamen gefunden, bzw. Fehler beim Ermitteln von Dateinamen
=	**98**		Nicht alle Dateien wurden bearbeitet
=	**999**		Fehler bei Speicheranforderung
3 →	**PARAM**	**CL*n***	Bereich mit Parametern
4 →	**PARLEN**	**F**	L nge des Parameterbereichs
=	**0**		keine Parameter vorhanden
>	**0**		Parameter vorhanden

Hinweis: Die Parameter m ssen in der gleichen Weise wie beim Dienstprogramm geschrieben werden.

F r Parameter sind nur Gro buchstaben zul ssig.

Beispiel f r den Aufruf von FLAMUP in COBOL:

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID. MUSTER.
*
* MUSTER FUER DEN AUFRUF VON FLAMUP
*
 ENVIRONMENT DIVISION.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 77 FLAMID          PIC S9(8) COMP SYNC.
 77 RETCO           PIC S9(8) COMP SYNC.
 77 PARAM           PIC X(80)
                         VALUE"C,MAXB=2"
 77 PARLEN          PIC S9(8) COMP SYNC VALUE 8.
```

```
*
 PROCEDURE DIVISION.
*
```

CALL "FLAMUP" USING FLAMID, RETCO, PARAM, PARLEN.

```
*
     STOP RUN.
```

Beispiel f r den Aufruf von FLAMUP in ASSEMBLER:

```
MUSTER    CSECT
          *
          *
          *
*
*   FLAMUP AUFRUFEN
*
          LA      1,FLAMUPAR
          L       15,=V(FLAMUP)
          BALR    14,15
          *
          *
          *
*
*   PARAMETER FUER FLAMUP
*
FLAMUPAR  DC      A(FLAMID)
          DC      A(RETCO)
          DC      A(PARAM)
          DC      A(X'80000000'+PARLEN)
*
FLAMID    DS      F
RETCO     DS      F
PARAM     DC      C'C,MODE=ADC'
PARLEN    DC      F'10'
*
*
*   SAVEAREA
*
SAVEAREA  DS      18F
          END
```

Registerbelegung f r ASSEMBLER:

→ R1: Adresse der Parameterliste
→ R13: zeigt auf Sicherstellungsbereich (18 Worte)
→ R14: enth lt die R cksprungadresse
→ R15: enth lt die Aufrufadresse

Beispiel f r den Aufruf von FLAMUP in C ++:

```
// an example for calling flamup from C++

//

// set linkage convention

extern "OS" void FLAMUP(void **,long *,char
*,long *);

int main()

{

 void *flamid;

 long retco;

 long parlen=8;

 char param[10]="C,MO=ADC";

 FLAMUP(&flamid,&retco,param,&parlen);

 return 0;

}
```

3.3 Satzschnittstelle FLAMREC

FLAMREC besteht aus einer Reihe von Unterprogrammen, die von allen Programmiersprachen wie COBOL, FORTRAN usw., sowie ASSEMBLER aufgerufen werden k nnen. Bis auf die Schl sselbeschreibung sind alle Parameter durch elementare Datentypen (INTEGER, STRING) dargestellt. Es werden bewusst keine Kontrollbl cke aufgebaut, so dass keine Ausrichtungsprobleme aufkommen und ein Kopieren von Parameterwerten vor und nach dem Funktionsaufruf berfl ssig ist. Nur die Schl sselbeschreibung ist im Interesse einer Abk rzung der Parameterliste als Struktur realisiert.

Alle Parameterlisten beginnen einheitlich mit einer Kennung, die zur eindeutigen Identifikation der Komprimatsdatei zwischen FLMOPN und FLMCLS dient, gefolgt von einem Returncode, der zur R ckmeldung der erfolgreichen Durchf hrung bzw. eines m glichen Fehlers dient.

Die Bearbeitung einer Komprimatsdatei beginnt immer mit der Funktion FLMOPN, in der die Zuordnung des Programms zur Komprimatsdatei erfolgt und die Verarbeitungsart festgelegt wird. Nach einem erfolgreichen ffnen ist die Bearbeitung immer mit FLMCLS abzuschlie en.

Es werden von der Satzschnittstelle keine Meldungen erzeugt.

Bei bergabe von Originals tzen enth lt der Parameter RECORD immer die Nettodaten ohne irgendwelche L ngenfelder oder Satztrenner bzw. der RECPTR zeigt auf ein Feld mit diesem Inhalt. Der Parameter RECLEN enth lt immer die L nge der Nettodaten.

COBOL-Programme k nnen mit der Option 'NODYNAM' bersetzt werden. Dadurch werden die FLAM-Module statisch mit dem aufrufenden Programm gelinkt (entspricht der V-Konstanten in ASSEMBLER). Dann sollte der FLAM-Modul FLAMREC explizit beim Binden angegeben werden.

Beispiel f r den Aufruf von FLMOPF in COBOL:

```
IDENTIFICATION DIVISION.
PROGRAM-ID. MUSTER.
*
*  MUSTER FUER DEN AUFRUF VON FLMOPF
*
ENVIRONMENT DIVISION
DATA DIVISION.
WORKING-STORAGE SECTION.
77  FLAMID            PIC S9(8) COMP SYNC.
77  RETCO             PIC S9(8) COMP SYNC.
77  VERSION           PIC S9(8) COMP SYNC.
77  FLAMCODE          PIC S9(8) COMP SYNC.
77  COMPMODE          PIC S9(8) COMP SYNC.
77  MAXBUFF           PIC S9(8) COMP SYNC.
77  HEADER            PIC S9(8) COMP SYNC.
77  MAXREC            PIC S9(8) COMP SYNC.
77  BLKMODE           PIC S9(8) COMP SYNC.
77  EXK20             PIC X(8)  VALUE SPACES.
77  EXD20             PIC X(8)  VALUE SPACES.
01  KEYDESC.
    05 KEYFLAGS       PIC S9(8) COMP SYNC.
    05 KEYPARTS       PIC S9(8) COMP SYNC.
    05 KEYELEM        OCCURS 8 TIMES.
       10 KEYPOS      PIC S9(8) COMP SYNC.
       10 KEYLEN      PIC S9(8) COMP SYNC.
       10 KEYTYPE     PIC S9(8) COMP SYNC.
*
PROCEDURE DIVISION.
*
    CALL "FLMOPF" USING  FLAMID, RETCO, VERSION
         FLAMCODE, COMPMODE, MAXBUFF, HEADER,
```

```
          MAXREC, KEYDESC, BLKMODE, EXK20,
EXD20.
*
          *
          *
          *
```

Beispiel f r den Aufruf von FLMOPF in ASSEMBLER:

```
MUSTER   CSECT
         *
         *
         *
*
*  STANDARDWERTE EINSTELLEN
*
         LA    0,0
         ST    0,COMPMODE    COMPMODE = CX8
         ST    0,HEADER      HEADER = NO
         ST    0,BLKMODE     BLKMODE = YES
         LA    0,255
         ST    0,MAXREC      MAXRECORDS = 255
         L     0,=F'32768'
         ST    0,MAXBUFF     MAXBUFFER = 32768
         LA    0,1
         ST    0,KEYPARTS    KEYPARTS = 1
         ST    0,KEYPOS1     KEYPOS1 = 1
         LA    0,0
         ST    0,KEYFLAGS    = NO DUPLICATE KEY
         ST    0,KEYTYPE1    = ABDRUCKBAR
         LA    0,8
         ST    0,KEYLEN1     KEYLEN1 = 8
         MVI   EXK20,C' '    KEIN EXK20
         MVI   EXD20,C' '    KEIN EXD20

*
*  PARAMETERLISTE FUER FLMOPF AUFBAUEN
*
         LA    15,VERSION
         ST    15,ARVERSIO
```

```
         LA    15,CODE
         ST    15,ARCODE
         LA    15,COMPMODE
         ST    15,ARCOMPMO
         LA    15,MAXBUFF
         ST    15,ARMAXBUF
         LA    15,HEADER
         ST    15,ARHEADER
         LA    15,MAXREC
         ST    15,ARMAXREC
         LA    15,KEYDESC
         ST    15,ARKYDESF
         LA    15,BLKMODE
         ST    15,ARBLKMOD
         LA    15,EXK20
         ST    15,AREXK20
         LA    15,EXD20
         ST    15,AREXD20
*
*  FLMOPF AUFRUFEN
*
         LA    1,RECPAR
         L     15,=V(FLMOPF)
         BALR  14,15
```

Parameterlisten f r FLAMREC

```
*
*  PARAMETERLISTE FUER FLMOPN
*
RECPAR   DS    0A
ARFLAMID DS    A   ADRESSE FLAMID
ARETCO   DS    A   ADRESSE RETCO
```

```
AREST    DS    0F
ARLAST   DS    A   ADRESSE LASTPAR
ARMODE   DS    A   ADRESSE MODE
ARLINK   DS    A   ADRESSE LINKNAME (DDNAME)
ARSTATIS DS    A   ADRESSE STATIS
*
*  PARAMETER FUER FLMOPD
*
         ORG   ARMODE
ARNLEN   DS    A   ADRESSE NAMELEN
ARNAME   DS    A   ADRESSE FILENAME
ARDSORG  DS    A   ADRESSE DSORG
ARECFORM DS    A   ADRESSE REFORM
ARMAXSIZ DS    A   ADRESSE MAXSIZE
ARECDELI DS    A   ADRESSE RECDELIM
ARKYDESD DS    A   ADRESSE KEYDESC
ARBLKSIZ DS    A   ADRESSE BLKSIZE
ARCLOSDI DS    A   ADRESSE CLOSDISP
ARDEVICE DS    A   ADRESSE DEVICE

*
*  PARAMETER FUER FLMOPF
*
         ORG   AREST
ARVERSIO DS    A   ADRESSE VERSION
ARCODE   DS    A   ADRESSE CODE
ARCOMPMO DS    A   ADRESSE COMPMODE
ARMAXBUF DS    A   ADRESSE MAXBUFFER
ARHEADER DS    A   ADRESSE HEADER
ARMAXREC DS    A   ADRESSE MAXREC
ARKYDESF DS    A   ADRESSE KEYDESC
ARBLKMOD DS    A   ADRESSE BLKMODE
AREXK20  DS    A   ADRESSE EXK20
```

```
AREXD20  DS    A   ADRESSE EXD20
*
*  PARAMETER FUER FLMCLS
*
         ORG   AREST
ARCPUTIM DS    A   ADRESSE CPUTIME
ARECORDS DS    A   ADRESSE RECORDS
ARBYTES  DS    A   ADRESSE BYTES
ARBYTOFL DS    A   ADRESSE BYTEOFL
ARCMPREC DS    A   ADRESSE CMPRECS
ARCMPBYT DS    A   ADRESSE CMPBYTES
ARCBYOFL DS    A   ADRESSE CBYTEOFL
*
*  PARAMETER FUER FLMGET, FLMLOC UND FLMPUT
*
         ORG   AREST
ARECLEN  DS    A   ADRESSE RECLEN
ARECPTR  DS    A   ADRESSE RECORD (RECPTR BEI
LOCATE)
ARBUFLEN DS    A   ADRESSE BUFLEN
*
*  PARAMETER FUER FLMPOS
*
         ORG   AREST
ARPOS    DS   A   ADRESSE POSITION


*
*  PARAMETER FUER FLMGHD UND FLMPHD
*
         ORG   AREST
ARHNAML  DS   A    ADRESSE NAMLENE
ARHNAME  DS   A    ADRESSE FILENAME
ARHFCBT  DS   A    ADRESSE DATEIFORMAT
```

```
ARHRECF  DS   A    ADRESSE SATZFORMAT
ARHRECS  DS   A    ADRESSE SATZLAENGE
ARHRECD  DS   A    ADRESSE RECDELIM
ARHKEYD  DS   A    ADRESSE KEYDESC
ARHBLKS  DS   A    ADRESSE BLOCKLAENGE
ARHPRCTR DS   A    ADRESSE
VORSCHUBSTEUERZEICHEN
ARHSYST  DS   A    ADRESSE BETRIEBSSYSTEM - ID
APHLAST  DS   A    ADRESSE LASTPAR nur FLMPHD
         ORG
*
*  PARAMETERWERTE FUER FLAMREC
*
RETCO    DS   F    RETURNCODE
FLAMID   DS   F    FLAMFILE-ID
LASTPAR  DS   F    ENDE DER PARAMETEREINGABE
OPENMODE DS   F    OPENMODE
POSITION DS   F    RELATIVE POSITION
ABSPOS   DS   F    ABSOLUTE POSITION
*
NAMELEN  DS   F    LAENGE DATEINAMEN FLAMFILE
FILENAME DS   CL54 DATEINAMEN DER FLAMFILE
DSORG    DS   F    DSORG
RECFORM  DS   F    RECFORM
MAXSIZE  DS   F    MAXSIZE
RECDELIM DS   XL4  RECDELIM
KEYSIZE  DS   F    LAENGE ALLER TEILSCHLUESSEL
BLKSIZE  DS   F    BLKSIZE
CLOSDISP DS   F    CLOSDISP
DEVICE   DS   F    DEVICE
*
VERSION  DS   F    FLAM-VERSION
CODE     DS   F    FLAMCODE
```

```
COMPMODE DS   F    COMPMODE
MAXBUFF  DS   F    MAXBUFFER
HEADER   DS   F    HEADER
MAXREC   DS   F    MAXRECORDS
BLKMODE  DS   F    BLKMODE
EXK20    DS   CL8  EXK20
EXD20    DS   CL8  EXD20
*
CPUTIME  DS   F    CPUZEIT IN MILLISEKUNDEN
ELATIME  DS   F    LAUFZEIT IN MILLISEKUNDEN
RECORDS  DS   F    ANZAHL ORIGINALSAETZE
BYTES    DS   F    ANZAHL ORIGINALBYTES
BYTEOFL  DS   F    UEBERLAUFZAEHLER FUER
*                  ORIGINALBYTES
CMPRECS  DS   F    ANZAHL KOMPRIMATSSAETZE
CMPBYTES DS   F    ANZAHL KOMPRIMATSBYTES
CBYTEOFL DS   F    UEBERLAUFZAEHLER FUER
*                  KOMPRIMATSBYTES
*
*  SCHLUESSELBESCHREIBUNG
*
KEYDESC  DS    0F  KEYFLAGS DS    F
KEYPARTS DS    F   ANZAHL SCHLUESSELTEILE
KEYPOS1  DS    F   ERSTES BYTE DES ERSTEN TEILS
KEYLEN1  DS    F   LAENGE DES ERSTEN TEILS
KEYTYPE1 DS    F   DATENTYP DES ERSTEN TEILS
KEYPOS2  DS    F
KEYLEN2  DS    F
KEYTYPE2 DS    F
KEYPOS3  DS    F
KEYLEN3  DS    F
KEYTYPE3 DS    F
KEYPOS4  DS    F
```

```
KEYLEN4  DS    F

KEYTYPE4 DS    F

KEYPOS5  DS    F

KEYLEN5  DS    F

KEYTYPE5 DS    F

KEYPOS6  DS    F

KEYLEN6  DS    F

KEYTYPE6 DS    F

KEYPOS7  DS    F

KEYLEN7  DS    F

KEYTYPE7 DS    F

KEYPOS8  DS    F    ERSTES BYTE DES LETZTEN
TEILS

KEYLEN8  DS    F    LAENGE DES LETZTEN TEILS

KEYTYPE8 DS    F    DATENTYP DES LETZTEN TEILS

*

RECLEN   DS    F

RECPTR   DS    A

*

*   SAVEAREA

*

SAVEAREA DS    18F

         *

         *

         *

         END
```

3.3.1 Funktion FLMOPN

Die Funktion FLMOPN muss als erste aufgerufen werden. Die Zuordnung zwischen Programm und Komprimatsdatei und die Verarbeitungsart werden festgelegt.

Parameter:

1 ← **FLAMID** **F** Kennung. Muss bei allen nachfolgenden Aufrufen unver- ndert bergeben werden

2 ← **RETCO** **F** Returncode (siehe auch Kapitel 8.4)

=	**0**	Kein Fehler
=	**-1**	Fehler bei Speicheranforderung
=	**10**	Datei ist keine FLAMFILE
=	**11**	FLAMFILE Formatfehler
=	**12**	Satzl ngenfehler
=	**13**	Dateil ngenfehler
=	**14**	Checksummenfehler
=	**20**	Unzul ssiger OPENMODE
=	**21**	Unzul ssige Gr e des Matrixpuffers
=	**22**	Unzul ssiges Kompressionsverfahren
=	**23**	Unzul ssiger Code in FLAMFILE
=	**24**	Unzul ssiger MAXRECORDS-Parameter
=	**25**	Unzul ssige Satzl nge
=	**30**	FLAMFILE ist leer
=	**37**	Unzul ssige Schl sselposition (ungleich 1)
=	**40**	Modul oder Tabelle kann nicht geladen werden
=	**41**	Modul kann nicht aufgerufen werden
=	**42**	Modul kann nicht entladen werden
=	**43 - 49**	Fehlerabbruch durch Exit
=	**52**	unzul ssige doppelte Schl ssel in der FLAMFILE
=	**57**	unzul ssige Teilkomprimatsl nge
=	**x'F00000XX'**	FLAM-Fehlercode aus FLAMFIO f r FLAMFILE
	x' 1E' = 30	FLAMFILE ist leer
	x' 1F' = 31	FLAMFILE nicht zugeordnet
	x' 20' = 32	Unzul ssiger OPENMODE
	x' 21' = 33	Ung ltiger Dateityp
	x' 22' = 34	Ung ltiges Satzformat
	x' 23' = 35	Ung ltige Satzl nge
	x' 24' = 36	Ung ltige Blockl nge
	x' 26' = 38	Ung ltige Schl ssell nge
	x' 27' = 39	Ung ltiger Dateiname
	x' 28' = 40	Ein FLAM-Modul konnte nicht geladen werden.
=	**x'FFXXXXXX'**	DMS-Fehlercode aus FLAMFIO f r FLAMFILE

3 →	**LASTPAR**	**F**	Ende der Parameter bergabe f r OPEN
	= 0		Keine weitere Parameter bergabe
	= sonst		Weiterer Funktionsaufruf mit FLMOPD bzw. FLMOPF
4 →	**OPENMODE**	**F**	Der Openmode bestimmt die Arbeitsweise
	= 0		INPUT = FLAMFILE lesen-DEKOMPRIMIEREN
	= 1		OUTPUT = FLAMFILE schreiben-KOMPRIMIEREN
	= 2		INOUT (mit Schl ssel und sequentiell lesen und ndern)
5 →	**DDNAME**	**CL8**	Symbolischer Dateiname mit Leerzeichen aufgef llt
6 →	**STATIS**	**F**	Statistik einschalten oder nicht
	= 0		Keine Statistik
	= 1		Statistik-Daten sammeln und mit FLMCLS an den Benutzer bergeben

3.3.2 Funktion FLMOPD

Die Funktion FLMOPD beschreibt spezielle Dateieigenschaften der FLAMFILE. Falls FLMOPD benutzt wird, muss die Funktion als zweite nach FLMOPN aufgerufen werden. Anderenfalls werden die im Funktionsaufruf angegebenen Standardwerte benutzt. Eine Generierung ist nicht vorgesehen.

Parameter:

1 →	**FLAMID**	**F**	Kennung
2 ←	**RETCO**	**F**	Returncode
	= 0		Kein Fehler
	= -1		Ung ltige Kennung, unzul ssiger Aufruf (z.B. LASTPAR= 0 bei FLMOPN)
	= sonst		Weitere Returncodes siehe FLMOPN
3 →	**LASTPAR**	**F**	Ende der Parameter bergabe f r OPEN
	= 0		Keine weitere Parameter bergabe
	= sonst		Weiterer Funktionsaufruf mit FLMOPF
4 ↔	**NAMELEN**	**F**	L nge des Dateinamens bzw. des Bereichs f r den Dateinamen (STANDARD = 0)
5 ↔	**FILENAME**	**CL*n***	Dateiname der FLAMFILE. Dateiname wird zur ckgegeben, wenn er nicht angegeben ist. (Erstes Zeichen ist Leerzeichen)
6 ↔	**DSORG**	**F**	Dateiformat der FLAMFILE
	= 0; 8; 16		Sequentiell (DTFSD, DTFMT, SAM-ESDS)
	= 1; 9; 17		Indexsequentiell (KSDS)
	= 2; 10		Relativ (RRDS)
7 ↔	**RECFORM**	**F**	Satzformat der FLAMFILE
	= 0; 8; 16		VAR; VARBLK,SPNBLK
	= 1; 9; 17		FIX; FIXBLK
	= 2; 10; 18		UNDEF
8 ↔	**MAXSIZE**	**F**	Maximale Satzl nge der FLAMFILE, zul ssige Werte: 80 - 32760. Bei CX7 ist f r die FLAMFILE nur eine maximale Satzl nge von 4096 zul ssig. (512 = STANDARD)
9 ↔	**RECDELIM**	**XL*n***	Satztrenner (wird z.Zt. nicht unterst tzt)

10 →	**KEYDESC**	**STRUCT**	Schl sselbeschreibung f r die Originals tze (es muss die Adresse der Struktur bergeben werden)
←			Schl sselbeschreibung der FLAMFILE
	KEYFLAGS	**F**	Option
	=	**0**	Keine doppelten Schl ssel (STANDARD)
	=	**1**	Doppelte Schl ssel zul ssig
	KEYPARTS	**F**	Anzahl der Schl sselteile
	=	**1 bis 8**	(STANDARD= 0, keine Schl ssel)
	KEYPOS1	**F**	Byteposition des ersten Teilschl ssels
	=	**1 - 32759**	(STANDARD = 1)
	KEYLEN1	**F**	L nge des ersten Teilschl ssels
	=	**1 - 255**	(STANDARD = 8)
	KEYTYPE1	**F**	Datentyp des ersten Teilschl ssels
	=	**0**	Abdruckbare Zeichen
	=	**1**	Bin rwerte (STANDARD)
	.		
	.		
	.		
	KEYPOS8	**F**	Byteposition des achten Teilschl ssels
	=	**1 - 32759**	(STANDARD = 1)
	KEYLEN8	**F**	L nge des achten Teilschl ssels
	=	**1 - 255**	(STANDARD = 8)
	KEYTYPE8	**F**	Datentyp des achten Teilschl ssels
	=	**0**	Abdruckbare Zeichen
	=	**1**	Bin rwerte (STANDARD)
11 ↔	**BLKSIZE**	**F**	Blocksize
	=	**0**	ungeblockt
	=	**80 - 32760**	
12 ↔	**CLOSDISP**	**F**	Art der Close-Bearbeitung
	=	**0**	REWIND (STANDARD)
	=	**1**	UNLOAD
	=	**2**	LEAVE
13 ↔	**DEVICE**	**F**	Ger tetyp
	=	**0; 8; 16**	Platte bzw. nicht bekannt (STANDARD)
	=	**1; 9; 17**	Magnetband
	=	**2; 10; 18**	Diskette
	=	**3; 11; 19**	Streamer
	=	**7; 15; 23**	Benutzer

FLAM errechnet sich aus der Schl sselbeschreibung der Originaldatei eine optimale Schl ssell nge. Diese ist bei bin ren Komprimaten 1 Byte l nger als die Summe der Originalschl ssel, bei abdruckbarer Komprimatssyntax

(MODE=CX7) werden 2 Byte erg nzt. Die Schl sselposition ist stets 1. Wird die KSDS-FLAMFILE mittels IDCAMS selbst angelegt, so sollten o.a. Angaben ber cksichtigt werden. Eine zu geringe Schl ssell nge f hrt zu Performaceverlust bei der weiteren Verarbeitung.

Werden doppelte Schl ssel f r die Originaldatens tze zugelassen (KEYFLAGS=1), werden 2 (bzw. 4) Byte erg nzt. Damit werden im Sinne von VSAM f r die FLAMFILE keine S tze mit doppeltem Schl ssel erzeugt.

3.3.3 Funktion FLMOPF

Die Funktion FLMOPF definiert die Komprimatseigenschaften. FLMOPF kann als zweite Funktion nach FLMOPN oder als dritte nach FLMOPD aufgerufen werden.

Parameter:

1 →	**FLAMID**	**F**	Kennung
2 ←	**RETCO**	**F**	Returncode
	=	**0**	Kein Fehler
	=	**-1**	Ung ltige Kennung, unzul ssiger Aufruf (z.B.
LASTPAR=0			
			bei FLMOPN oder FLMOPD)
	=	**40**	Exit konnte nicht geladen werden
	=	**43-49**	Fehlerabbruch durch Exit
	=	**sonst**	Weitere Returncodes siehe FLMOPN
3 ↔	**VERSION**	**F**	FLAM-Version
	=	**100**	Version 1 / 6020
	=	**101**	Version 1 / 6035
	=	**200**	Version 2
	=	**300**	Version 3
	=	**400**	Version 4
4 ↔	**FLAMCODE**	**F**	Zeichencode der FLAMFILE
	=	**0**	EBCDIC
	=	**1**	ASCII
5 ↔	**COMPMODE**	**F**	Kompressionsverfahren
	=	**0**	CX8 (STANDARD)
	=	**1**	CX7
	=	**2**	VR8
	=	**3**	ADC
	=	**5**	NDC
6 ↔	**MAXBUFF**	**F**	Gr e des Matrixpuffers in BYTES. Es ist jeder positive Wert zul ssig, es wird der tats chlich benutzte Wert zur ckgegeben (STANDARD = 65536)
7 ↔	**HEADER**	**F**	FILEHEADER erzeugen bzw. vorhanden
	=	**0**	Kein Fileheader erzeugen bzw. vorhanden
	=	**1**	Fileheader erzeugen bzw. vorhanden

Nr.	Parameter	Format	Beschreibung
8 →	**MAXREC**	**F**	Maximale Satzanzahl in der Matrix (STANDARD = 255)
	= 1- 4095		(STANDARD = 255 f r CX7, CX8 und VR8) (STANDARD = 4095 f r ADC)
9 ↔	**KEYDESC STRUCT**		Schl sselbeschreibung f r die Originals tze (es muss die Adresse der Struktur bergeben werden)
	KEYFLAGS	**F**	Option
	= 0		Keine doppelten Schl ssel (STANDARD)
	= 1		Doppelte Schl ssel zul ssig
	KEYPARTS	**F**	Anzahl der Schl sselteile
	= 1 bis 8		(STANDARD =0, keine Schl ssel)
	KEYPOS1	**F**	Byteposition des ersten Teilschl ssels
	= 1 - 32759		(STANDARD = 1)
	KEYLEN1	**F**	L nge des ersten Teilschl ssels
	= 1 - 255		(STANDARD = 8)
	KEYTYPE1	**F**	Datentyp des ersten Teilschl ssels
	= 0		Abdruckbare Zeichen
	= 1		Bin rwerte (STANDARD)
	.		
	.		
	.		
	KEYPOS8	**F**	Byteposition des achten Teilschl ssels
	= 1 - 32759		(STANDARD = 1)
	KEYLEN8	**F**	L nge des achten Teilschl ssels
	= 1 - 255		(STANDARD = 8)
	KEYTYPE8	**F**	Datentyp des achten Teilschl ssels
	= 0		Abdruckbare Zeichen
	= 1		Bin rwerte (STANDARD)
10 →	**BLKMODE**	**F**	Geblockte bzw. ungeblockte Ausgabe f r sequentielle Komprimatsdateien
	= 0		Ungeblockt (in einem Komprimatssatz sind nur Daten aus der gleichen Matrix)
	= 1		Geblockt (STANDARD) (in einem Komprimatssatz k nnen sich Daten von mehreren Matrizen befinden)
11 →	**EXK20**	**CL8**	Spaces oder Name des Benutzerausgangs f r die Komprimatsausgabe (STANDARD = SPACES)
12 →	**EXD20**	**CL8**	Spaces oder Name des Benutzerausgangs f r die Komprimatseingabe (STANDARD = SPACES)
←			Wenn beim Dekomprimieren der Exit *STREAM automatisch aktiviert wird.

3.3.4 Funktion FLMCLS

Mit der Funktion FLMCLS wird der Zugriff auf die Satzschnittstelle beendet. Bei der Komprimierung wird noch die letzte Matrix komprimiert, das Komprimat auf die FLAMFILE geschrieben und dann die FLAMFILE geschlossen. Beim Dekomprimieren wird nur die FLAMFILE geschlossen, falls noch vorhanden, werden restliche Originals tze nicht mehr   bergeben.

Falls mit FLMOPN angefordert (STATIS=1), werden die Statistikinformationen mit   bergeben.

Parameter:

1 →	**FLAMID**	**F**	Kennung
2 ←	**RETCO**	**F**	Returncode
	= 0		Kein Fehler
	= -1		Ung ltige Kennung
	= 43 - 49		Fehlerabbruch durch Exit
	= x'FFXXXXXX'		DMS-Fehlercode

Die folgenden Parameter werden nur bei eingeschalteter Statistik benutzt.

3 ←	**CPUTIME**	**F**	CPU-Zeit in Millisekunden in fremden Prozessen
4 ←	**RECORDS**	**F**	Anzahl Originals tze
5 ←	**BYTES**	**F**	Anzahl Originalbytes
6 ←	**BYTEOFL**	**F**	berlaufz hler f r Originalbytes
7 ←	**CMPRECS**	**F**	Anzahl Komprimatss tze
8 ←	**CMPBYTES**	**F**	Anzahl Komprimatsbytes
9 ←	**CMPBYOFL**	**F**	berlaufz hler f r Komprimatsbytes

Bei extrem gro en Komprimatsdateien (gr er als 4 Gigabytes) reichen die Bytez hler von einem Wort nicht mehr aus. Zu diesem Zweck sind die   berlaufz hler vorgesehen. Damit k nnen die Z hler auf ein Doppelwort erweitert werden:

```
01 BYTEFELD.
    05 BYTEOFL              PIC 9(8) COMP SYNC.
    05 BYTES                PIC 9(8) COMP SYNC.
```

```
01 BYTECNT  REDEFINES BYTEFELD PIC S9(18) COMP
SYNC.
```

3.3.5 Funktion FLMDEL

Mit der Funktion FLMDEL kann der zuletzt gelesene Originalsatz aus einer indexsequentiellen FLAMFILE gel scht werden.

Parameter:

1 →	**FLAMID**	**F**	Kennung
2 ←	**RETCO**	**F**	Returncode
	=	**0**	Kein Fehler
	=	**-1**	Ung ltige Kennung oder Funktion unzul ssig
	=	**5**	Kein aktueller Satz vorhanden
	=	**43 - 49**	Fehlerabbruch durch Exit
	=	**x'FFXXXXXX'**	DMS-Fehlercode siehe bei FLMOPN

3.3.6 Funktion FLMFKY

Mit FLMFKY (Find Key) kann in einer indexsequentiell organisierten FLAMFILE ein Satz der Originaldatei gesucht werden, dessen Schl ssel einem vorgegebenen Schl sselwert entspricht oder gr er ist. Der Vorgabewert kann generisch sein, d.h. nicht alle Stellen des Schl sselwertes m ssen eindeutig angegeben werden. Der gefunden Satz ist der n chste zu verarbeitende Satz.

Wird mit FLMFKY kein Satz gefunden, bleibt die alte Position erhalten.

Parameter:

1 →	**FLAMID**	**F**	Kennung
2 ←	**RETCO**	**F**	Returncode
	= 0		Kein Fehler
	= -1		Ung ltige Kennung oder Funktion unzul ssig
	= 5		Schl ssel nicht vorhanden
	= sonst		siehe Funktion FLMGET
3 →	**KEYLEN**	**F**	Schl ssell nge Es enth lt die Anzahl signifikanter Bytes im vorgegebenen Schl sselwert. Es kann kleiner sein als die Schl ssell nge. In diesem Fall wird bei dem im Argument checkmod angegebenen logischen Vergleich nur die hier bergebene L nge ber cksichtigt.
4 →	**RECORD**	**XL*n***	Satzpuffer mit Suchschl ssel
5 →	**CHECKMOD**	**F**	Vergleichsart
	= 0		gleich
	= 1		gr er oder gleich
	= 2		gr er

3.3.7 Funktion FLMFLU

Mit dieser Funktion wird die aktuelle FLAM-Matrix abgeschlossen. Falls mit FLMOPN angefordert (STATIS=1), werden die Statistikinformationen mit bergeben. Bei der Komprimierung wird der Matrixinhalt sofort komprimiert und weggeschrieben, bei der Dekomprimierung die n chste Matrix dekomprimiert.

Parameter:

1 →	**FLAMID**	**F**	Kennung
2 ←	**RETCO**	**F**	Returncode
	= 0		Kein Fehler
	= -1		Ung ltige Kennung
	= 43 - 49		Fehlerabbruch durch Exit
	= x'FFXXXXXX'		DMS-Fehlercode

Die folgenden Parameter werden nur bei eingeschalteter Statistik benutzt.

3 ←	**CPUTIME**	**F**	CPU-Zeit in Millisekunden in fremden Prozessen
4 ←	**RECORDS**	**F**	Anzahl Originals tze
5 ←	**BYTES**	**F**	Anzahl Originalbytes
6 ←	**BYTEOFL**	**F**	berlaufz hler f r Originalbytes
7 ←	**CMPRECS**	**F**	Anzahl Komprimatss tze
8 ←	**CMPBYTES**	**F**	Anzahl Komprimatsbytes
9 ←	**CMPBYOFL**	**F**	berlaufz hler f r Komprimatsbytes

Bei extrem gro en Komprimatsdateien (gr er als 4 Gigabytes) reichen die Bytez hler von einem Wort nicht mehr aus. Zu diesem Zweck sind die berlaufz hler vorgesehen. Damit k nnen die Z hler auf ein Doppelwort erweitert werden:

```
01 BYTEFELD.
    05 BYTEOFL              PIC 9(8) COMP SYNC.
    05 BYTES                PIC 9(8) COMP SYNC.
01 BYTECNT  REDEFINES BYTEFELD PIC S9(18) COMP
SYNC.
```

3.3.8 Funktion FLMFRN

Mit FLMFRN (Find Record-Number) wird auf einen Satz mit in einer vorgegebenen Nummer in einer indexsequentiellen FLAMFILE positioniert. Diese Nummer entspricht der Satznummer der sequentiellen oder relativen Originaldatei. Der Satz ist der n chste zu verarbeitende Satz. Mit der Angabe checkmod = 1 oder 2 kann ber L cken und leere S tze positioniert werden.

Wird mit FLMFRN kein g ltiger Satz gefunden, bleibt die alte Position erhalten.

Parameter:

1 →	**FLAMID**	**F**	Kennung
2 ←	**RETCO**	**F**	Returncode
	= 0		Kein Fehler
	= -1		Ung ltige Kennung oder Funktion unzul ssig
	= 5		Ung ltige Position
	= sonst		siehe Funktion FLMGET
3 ↔	**RECNO**	**F**	Satznummer
	= 1		Dateianfang. Bei Checkmod=1,2 wird die tats chliche Satznummer zur ckgegeben
4 →	**CHECKMOD**	**F**	Vergleichsart
	= 0		Satz mit angegebener Nummer
	= 1		Satz mit angegebener Nummer, L cken und leere S tze berspringen
	= 2		Satz mit n chster Nummer, L cken und leere S tze ber-springen

3.3.9 Funktion FLMGET

Mit der Funktion FLMGET wird der jeweils n chste Originalsatz in sequentieller Folge gelesen. Es ist m glich, mit FLMGKY oder FLMPOS in einer Komprimatsdatei zu positionieren und danach sequentiell weiterzulesen. Die Daten werden dabei in den Satzpuffer des aufrufenden Programms bertragen (move Mode).

Parameter:

1 →	**FLAMID**	**F**	Kennung
2 ←	**RETCO**	**F**	Returncode
	= 0		Kein Fehler
	= -1		Ung ltige Kennung oder Funktion unzul ssig
	= 1		Satz wurde verk rzt, da Originalsatz l nger als BUFLEN
	= 2		END-OF-FILE wurde erreicht
	= 3		L cke bei relativer Datei gefunden
	= 6		Neue Datei beginnt; gegebenenfalls kann der neue Fileheader gelesen werden.
	= 7		Passwort fehlt
	= 11		FLAMFILE Formatfehler
	= 12		Satzl ngenfehler
	= 13		Dateil ngenfehler
	= 14		Checksummenfehler
	= 15		Ung ltige Satzl nge (negativ)
	= 29		Passwort ung ltig
	= 43 - 49		Fehlerabbruch durch Exit
	= 52		Zuviele oder unzul ssige doppelte Schl ssel
	= x'FFXXXXXX'		DMS-Fehlercode
	= sonst		siehe Kapitel 8.4
3 ←	**RECLEN**	**F**	Satzl nge in Bytes des bergebenen Satzes
4 ←	**RECORD**	**XL*n***	Originalsatz (Daten)
5 →	**BUFLEN**	**F**	L nge des verf gbaren Satzpuffers in Bytes

Hinweis:
Bei den Returncodes 2 und 6 wird kein Satz bergeben.

Bei Returncode 3 wird ein Satz der L nge 0 bergeben.

3.3.10 Funktion FLMGHD

Die Funktion FLMGHD (Get-File-Header) ist nur bei der Dekomprimierung zugelassen. Der Fileheader beschreibt das Dateiformat der Originals tze. Zwischen FLAM-OPEN (FLMOPN, FLMOPD, FLMOPF) und FLAM-CLOSE (FLMCLS) kann der Fileheader mit der Funktion FLMGHD jederzeit angefordert werden. Sind in der FLAMFILE mehrere Fileheader vorhanden (siehe FLMPHD), so wird mit FLMGHD jeweils der letzte von FLAM erkannte Fileheader bergeben. Der erste Fileheader steht normalerweise nach FLAM-OPEN (siehe FLMOPF HEADER=1) zur Verf gung. Erkennt FLAM weitere Fileheader, so wird dies dem Benutzer im Returncode (RETCO=6) von FLMGET bzw. FLMLOC kenntlich gemacht.

Parameter:

1 →	**FLAMID**	**F**	Kennung
2 ←	**RETCO**	**F**	Returncode
	= 0		Kein Fehler
	= -1		Ung ltige Kennung oder Funktion unzul ssig
3 ↔	**NAMLEN**	**F**	L nge des Dateinamens bzw. des Bereichs
	= 0		Dateiname nicht bekannt
4 ←	**FILENAME**	**CL*n***	Dateiname der Originaldatei
5 ←	**DSORG**	**F**	Dateiformat
	= 0		sequentiell
	= 1		indexsequentiell
	= 2		relativ
	= 3		Direktzugriff
	= 5		Bibliothek
	= 6		physikalisch
6 ←	**RECFORM**	**F**	Satzformat
	= 0; 8; 16 ...		**V:** VARIABEL; 8 = VARBLK; 16=SPNBLK
	= 1; 9; 17 ...		**F:** FIX; 9 = FIXBLK
	= 2; 10; 18 ...		**U:** UNDEFINED
	= 3; 11; 19 ...		**S:** STREAM; 11 = Texttrenner; 19 = L ngenfelder
7 ←	**RECSIZE**	**F**	Satzl nge
	= 0 bis 32760		
	RECFORM = V:		Maximale Satzl nge oder 0
	RECFORM = F:		Satzl nge

RECFORM = U: Maximale Satzl nge oder 0

RECFORM = S: L nge des Texttrenners bzw. L ngenfeldes

8 ←	**RECDELIM**	**XL*n***	Satztrenner	
9 ←	**KEYDESC STRUCT**		Schl sselbeschreibung	
	KEYFLAGS	**F**	Optionen	
	=	**0**	Keine doppelten Schl ssel	
	=	**1**	Doppelte Schl ssel erlaubt	
	KEYPARTS	**F**	Anzahl Schl sselteile	
	=	**0 bis 8**	0 = Kein Schl ssel vorhanden	
	KEYPOS1	**F**	Erstes Byte des ersten Teilschl ssels	
	=	**1 bis 32759**	Wert < = Satzl nge	
	KEYLEN1	**F**	L nge des ersten Teilschl ssels	
	=	**1 bis 255**		
	KEYTYP1	**F**	Datentyp des ersten Teilschl ssels	
	=	**0**	Abdruckbare Zeichen	
	=	**1**	Bin rwert	
	.			
	.			
	.			
	KEYPOS8	**F**	Erstes Byte des achten Teilschl ssels	
	=	**1 bis 32759**	Wert < = Satzl nge	
	KEYLEN8	**F**	L nge des achten Teilschl ssels	
	=	**1 bis 255**		
	KEYTYP8	**F**	Datentyp des achten Teilschl ssels	
	=	**0**	Abdruckbare Zeichen	
	=	**1**	Bin rwert	
10 ←	**BLKSIZE**	**F**	Blockl nge	
	=	**0**	ungeblockt	
	=	**1 bis 32760**		
11 ←	**PRCTRL**	**F**	Vorschubsteuerzeichen	
	=	**0**	keine	
	=	**1**	ASA-Steuerzeichen	
	=	**2**	maschinenspezifische Steuerzeichen	
12 ←	**SYSTEM**	**XL2**	Betriebssystem, in dem die FLAMFILE erstellt wurde	
	=	**x’0000’**	nicht bekannt	
	=	**x’0080’**	MS-DOS	
	=	**x’0101’**	IBM	MVS z/OS
	=	**x’0102’**	IBM	VSE
	=	**x’0103’**	IBM	VM
	=	**x’0104’**	IBM	DPPX/8100
	=	**x’0105’**	IBM	DPPX/370
	=	**x’0106’**	IBM	AIX

=	**x'02XX'**	UNISYS	
=	**x'0301'**	DEC	VMS
=	**x'0302'**	DEC	ULTRIX
=	**x'0401'**	SIEMENS	BS2000
=	**x'0402'**	SIEMENS	SINIX
=	**x'0403'**	SIEMENS	SYSTEM V
=	**x'0501'**	NIXDORF	886X
=	**x'0502'**	NIXDORF	TARGON
=	**x'06XX'**	WANG	
=	**x'07XX'**	PHILLIPS	
=	**x'08XX'**	OLIVETTI	
=	**x'09XX'**	TANDEM	
=	**x'0AXX'**	PRIME	
=	**x'0BXX'**	STRATUS	
=	**x'0E02'**	APPLE A/UX	
=	**x'11XX'**	INTEL	80286
=	**x'12XX'**	INTEL	80386
=	**x'13XX'**	INTEL	80486

3.3.11 Funktion FLMGKY

Mit der Funktion FLMG Original-KY kann der Benutzer einen Originalsatz ber einen Schl ssel aus einer indexsequentiellen FLAMFILE anfordern. Der Suchschl ssel muss im Satzbereich an der Schl sselposition eingetragen werden.

Parameter:

1 →	**FLAMID**	**F**	Kennung
2 ←	**RETCO**	**F**	Returncode
	=	**0**	Kein Fehler
	=	**-1**	Ung ltige Kennung oder Funktion unzul ssig
	=	**1**	Satz wurde verk rzt, da Originalsatz l nger als BUFLEN
	=	**5**	Schl ssel nicht vorhanden
	=	**7**	Passwort fehlt
	=	**11**	FLAMFILE Formatfehler
	=	**12**	Satzl ngenfehler
	=	**13**	Dateil ngenfehler
	=	**14**	Checksummenfehler
	=	**29**	Passwort ung ltig
	=	**40 - 78**	siehe Funktion FLMGET
	=	**x'FFXXXXXX'**	DMS-Fehlercode
3 ←	**RECLEN**	**F**	Satzl nge in Bytes des bergebenen Satzes
4 ↔	**RECORD**	**XL*n***	Originalsatz (Daten mit Schl ssel)
5 →	**BUFLEN**	**F**	L nge des verf gbaren Satzpuffers in Bytes

3.3.12 Funktion FLMGRN

Die Funktion FLMGRN (Get Record-Number) liest den durch die Satznummer vorgegebenen Originalsatz einer sequentiellen oder relativen Datei aus einer indexsequentiellen FLAMFILE.

Wird mit FLMGRN kein g ltiger Satz gefunden, ist die neue Position der n chste Satz oder Dateiende.

Parameter:

1 →	**FLAMID**	**F**	Kennung
2 ←	**RETCO**	**F**	Returncode
	= 0		Kein Fehler
	= -1		Ung ltige Kennung oder Funktion unzul ssig
	= 1		Satz wurde verk rzt, da Originalsatz l nger als BUFLEN
	= 2		END-OF-FILE wurde erreicht
	= 3		L cke bei relativer Datei gefunden
	= 5		Ung ltige Satznummer (0 bzw. negativ)
	= 6		Neue Datei beginnt; gegebenenfalls kann der neue Fileheader gelesen werden.
	= sonst		siehe Kapitel 8.4
3 ←	**RECLEN**	**F**	Satzl nge in Bytes des bergebenen Satzes
4 ←	**RECORD**	***XLn***	Originalsatz (Daten)
5 →	**BUFLEN**	**F**	L nge des verf gbaren Satzpuffers in Bytes
6 →	**RECNO**	**F**	Satznummer
	= 1		Dateianfang

Bei den Returncodes 2 und 6 wird kein Satz bergeben.

Bei Returncode 3 wird ein Satz der L nge =0 bergeben.

3.3.13 Funktion FLMGTR

Mit der Funktion FLMGTR (Get reverse) wird der vorherige Originalsatz in sequentieller Folge gelesen. Es ist m glich, mit FLMGKY oder FLMPOS in einer Komprimatsdatei zu positionieren und danach sequentiell zur ckzulesen. Die Daten werden dabei in den Satzpuffer des aufrufenden Programms bertragen (move Mode).

Parameter:

1 →	**FLAMID**	**F**	Kennung
2 ←	**RETCO**	**F**	Returncode
	= 0		Kein Fehler
	= -1		Ung ltige Kennung oder Funktion unzul ssig
	= 3		L cke bei relativer Datei gefunden
	= 5		Ung ltige Satznummer (0 bzw. negativ)
	= 6		Neue Datei beginnt; gegebenenfalls kann der neue Fileheader gelesen werden.
	= sonst		siehe Kapitel 8.4
3 ←	**RECLEN**	**F**	Satzl nge in Bytes des bergebenen Satzes
4 ←	**RECORD**	**XL*n***	Originalsatz (Daten)
5 →	**BUFLEN**	**F**	L nge des verf gbaren Satzpuffers in Bytes

Bei den Returncodes 2 und 6 wird kein Satz bergeben.

Bei Returncode 3 wird ein Satz der L nge =0 bergeben.

3.3.14 Funktion FLMGUH

Die Funktion FLMGUH (Get User Header) liest die Benutzerdaten aus dem Fileheader der FLAMFILE.

Parameter:

1 →	**FLAMID**	**F**	Kennung
2 ←	**RETCO**	**F**	Returncode
	= 0		Kein Fehler
	= -1		Ung ltige Kennung oder Funktion unzul ssig
3 ↔	**UATTRLEN**	**F**	L nge der Benutzerdaten in Bytes bzw. L nge des Bereichs
	= 0		Keine Daten vorhanden
	= 1 - 3500		Bei 8-Bit Komprimaten (CX8, VR8, ADC)
	= 1 - 1750		Bei 7-Bit Komprimaten (CX7)
4 ←	**UATTR**	**XL*n***	Benutzerdaten

Die Benutzerdaten werden so wiedergegeben, wie sie geschrieben werden, d.h. eine Code-Umsetzung eines File-Transfers hat hier keine Wirkung.

3.3.15 Funktion FLMIKY

Die Funktion FLMIKY erlaubt S tze ber einen Schl ssel in eine indexsequentielle FLAMFILE (VSAM-KSDS) einzuf gen.

Parameter:

1 →	**FLAMID**	**F**	Kennung
2 ←	**RETCO**	**F**	Returncode
	= 0		Kein Fehler
	= -1		Ung ltige Kennung oder Funktion unzul ssig
	= 5		Schl ssel bereits vorhanden
	= 15		Originalsatz ist gr er als 32763 Bytes
	= 16		Originalsatz ist gr er als Matrix - 4
	= 43		Fehlerabbruch durch Exit
	= 52		Zuviele oder unzul ssige doppelte Schl ssel
	= x'FFXXXXXX'		DMS-Fehlercode
3 →	**RECLEN**	**F**	Satzl nge (Datenl nge) in Bytes ohne Satzl ngenfeld
4 →	**RECORD**	**XL*n***	Originalsatz (Daten mit Schl ssel)

3.3.16 Funktion FLMLCR

Die Funktion FLMLCR ist quivalent zu FLMGTR (Lesen r ckw rts). Die Daten werden dabei jedoch nicht bertragen, sondern es wird nur ein Zeiger auf den Satz zur Verf gung gestellt (locate Mode).

Parameter:

1 →	**FLAMID**	**F**	Kennung
2 ←	**RETCO**	**F**	Returncode
	= 0		Kein Fehler
	= -1		Ung ltige Kennung oder Funktion unzul ssig
	= 2		Dateianfang wurde erreicht
	= sonst		siehe Kapitel 8.4
3 ←	**RECLEN**	**F**	Satzl nge in Bytes des bergebenen Satzes
4 ←	**RECPTR**	**A**	Satzadresse (Datenadresse)

Hinweis:

Bei den Returncodes 2 und 6 wird keine Satzadresse bergeben. Bei Returncode 3 wird die L nge 0 bergeben.

3.3.17 Funktion FLMLOC

Die Funktion FLMLOC ist quivalent zu FLMGET. Die Daten werden dabei jedoch nicht bertragen, sondern es wird nur ein Zeiger auf den Satz zur Verf gung gestellt (locate Mode).

Parameter:

1 →	**FLAMID**	**F**	Kennung
2 ←	**RETCO**	**F**	Returncode
	= 0		Kein Fehler
	= -1		Ung ltige Kennung oder Funktion unzul ssig
	= 2		END-OF-FILE wurde erreicht
	= sonst		siehe FLMGET
3 ←	**RECLEN**	**F**	Satzl nge in Bytes des bergebenen Satzes
4 ←	**RECPTR**	**A**	Satzadresse (Datenadresse)

Hinweis:

Bei den Returncodes 2 und 6 wird keine Satzadresse bergeben.

Bei Returncode 3 wird die L nge 0 bergeben.

3.3.18 Funktion FLMPHD

Die Funktion FLMPHD (Put-File-Header) ist nur bei der Komprimierung zugelassen. Der Fileheader beschreibt das Dateiformat der anschlie end bergebenen Originals tze. Werden mehrere Dateien in eine FLAMFILE komprimiert, so kann f r jede Datei ein Fileheader mit der Funktion FLMPHD bergeben werden. FLAM gibt diese Fileheaderinformationen auf Anforderung (FLMGHD) beim Dekomprimieren zur ck. Die Funktion FLMPHD ist nur erlaubt, wenn bei FLMOPF HEADER=1 angegeben wird.

Parameter:

1 →	**FLAMID**	**F**	Kennung
2 ←	**RETCO**	**F**	Returncode
	= 0		Kein Fehler
	= -1		Ung ltige Kennung oder Funktion unzul ssig
3 →	**NAMLEN**	**F**	L nge des Dateinamens
	= 0		Dateiname nicht bernehmen
4 →	**FILENAME**	**CL*n***	Dateiname der Originaldatei
5 →	**DSORG**	**F**	Dateiformat
	= 0; 8; 16 ...		sequentiell
	= 1; 9; 17 ...		indexsequentiell
	= 2; 10; 18 ...		relativ
	= 3; 11; 19 ...		Direktzugriff
	= 5; 13; 21 ...		Bibliothek
	= 6; 14; 22 ...		physikalisch
6 →	**RECFORM**	**F**	Satzformat
	= 0; 8; 16 ...		**V:** VARIABEL; 8 = VARBLK; 16 = SPNBLK
	= 1; 9; 17 ...		**F:** FIX ; 9 = FIXBLK
	= 2; 10; 18 ...		**U:** UNDEFINED
	= 3; 11; 19 ...		**S:** STREAM; 11 = Texttrenner; 19 = L ngenfelder
7 →	**RECSIZE**	**F**	Satzl nge
	= 0 bis 32760		
	RECFORM = V:		Maximale Satzl nge oder 0
	RECFORM = F:		Satzl nge
	RECFORM = U:		Maximale Satzl nge oder 0
	RECFORM = S:		L nge des Texttrenners bzw. L ngenfeldes
8 →	**RECDELIM**	**XL*n***	Satztrenner

9 →	**KEYDESC STRUCT**		Schl sselbeschreibung	
	KEYFLAGS	**F**	Optionen	
	=	**0**	Keine doppelten Schl ssel	
	=	**1**	Doppelte Schl ssel erlaubt	
	KEYPARTS	**F**	Anzahl Schl sselteile	
	=	**0 bis 8**	0 = Kein Schl ssel vorhanden	
	KEYPOS1	**F**	Erstes Byte des ersten Teilschl ssels	
	=	**1 bis 32759**	Wert < = Satzl nge	
	KEYLEN1	**F**	L nge des ersten Teilschl ssels	
	=	**1 bis 255**		
	KEYTYP1	**F**	Datentyp des ersten Teilschl ssels	
	=	**0**	Abdruckbare Zeichen	
	=	**1**	Bin rwert	
	.			
	.			
	.			
	KEYPOS8	**F**	Erstes Byte des achten Teilschl ssels	
	=	**1 bis 32759**	Wert < = Satzl nge	
	KEYLEN8	**F**	L nge des achten Teilschl ssels	
	=	**1 bis 255**		
	KEYTYP8	**F**	Datentyp des achten Teilschl ssels	
	=	**0**	Abdruckbare Zeichen	
	=	**1**	Bin rwert	
10 →	**BLKSIZE**	**F**	Blockl nge	
	=	**0**	ungeblockt	
	=	**1 bis 32760**		
11 →	**PRCTRL**	**F**	Vorschubsteuerzeichen	
	=	**0**	keine	
	=	**1**	ASA-Steuerzeichen	
	=	**2**	maschinenspezifische Steuerzeichen (MCC)	
12 →	**SYSTEM**	**XL2**	Betriebssystem	
	=	**x'0000'**	nicht bekannt	
	=	**x'0080'**	MS-DOS	
	=	**x'0101'**	IBM	OS-MVS MVS/XA MVS/ESA
	=	**x'0102'**	IBM	VSE/SP VSE/ESA
	=	**x'0103'**	IBM	VM/SP VM/XA VM/ESA
	=	**x'0104'**	IBM	DPPX/8100
	=	**x'0105'**	IBM	DPPX/370
	=	**x'0106'**	IBM	AIX
	=	**x'02XX'**	UNISYS	
	=	**x'0301'**	DEC	VMS
	=	**x'0302'**	DEC	ULTRIX

=	**x’0401’**	SIEMENS	BS2000
=	**x’0402’**	SIEMENS	SINIX
=	**x’0403’**	SIEMENS	SYSTEM V

	=	**x’0501’**	NIXDORF	886X
	=	**x’0502’**	NIXDORF	TARGON
	=	**x’06XX’**	WANG	
	=	**x’07XX’**	PHILLIPS	
	=	**x’08XX’**	OLIVETTI	
	=	**x’09XX’**	TANDEM	
	=	**x’0AXX’**	PRIME	
	=	**x’0BXX’**	STRATUS	
	=	**x’0E02’**	APPLE A/UX	
	=	**x’11XX’**	INTEL	80286
	=	**x’12XX’**	INTEL	80386
	=	**x’13XX’**	INTEL	80486

13 →	**LASTPAR**	**F**	Ende Parameter bergabe f r Fileheader
	= 0		keine weitere Parameter bergabe
	sonst		es soll ein Benutzerheader mit FLMPUH bergeben werden

3.3.19 Funktion FLMPKY

Die Funktion FLMPKY erlaubt S tze ber einen Schl ssel, in eine indexsequentielle FLAMFILE einzuf gen oder zu   ndern.

Parameter:

1 →	**FLAMID**	**F**	Kennung
2 ←	**RETCO**	**F**	Returncode
	= 0		Kein Fehler
	= -1		Ung ltige Kennung oder Funktion unzul ssig
	= 5		Schl ssel nicht erlaubt
	= 15		Originalsatz ist gr er als 32763 Bytes
	= 16		Originalsatz ist gr er als Matrix - 4
	= 43 - 49		Fehlerabbruch durch Exit
	= 52		Zuviele oder unzul ssige doppelte Schl ssel
	= x'FFXXXXXX'		DMS-Fehlercode
3 →	**RECLEN**	**F**	Satzl nge (Datenl nge) in Bytes ohne Satzl ngenfeld
4 →	**RECORD**	**XL*n***	Originalsatz (Daten mit Schl ssel)

3.3.20 Funktion FLMPOS

Mit FLMPOS kann in Komprimatsdateien positioniert werden.

Parameter:

1 →	**FLAMID**	**F**	Kennung
2 ←	**RETCO**	**F**	Returncode
	= 0		Kein Fehler
	= -1		Ung ltige Kennung oder Funktion unzul ssig
	= 5		Unzul ssige Position
	= sonst		siehe Kapitel 8.4
3 →	**POSITION**	**F**	Position
	= - MAXINT		Dateianfang (-2147483648 bzw. X’80000000’ oder -99999999)
	= + MAXINT		Dateiende (+2147483647 bzw. X’7FFFFFFF’oder +99999999)
	= - N		N S tze r ckw rts
	= + N		N S tze vorw rts
	= - 9999 9998		Zur ck zum Anfang der aktuellen Datei bzw. zum Anfang der vorherigen Datei in einer Sammeldatei
	= + 9999 9998		Anfang der n chsten Datei in einer Sammeldatei

Hinweis: Zur Zeit ist es nur m glich vorw rts zu positionieren.

3.3.21 Funktion FLMPUH

Die Funktion FLMPUH (Put User Header) schreibt Benutzerdaten in den Fileheader der FLAMFILE.

Parameter:

1 →	**FLAMID**	**F**	Kennung
2 ←	**RETCO**	**F**	Returncode
	= 0		Kein Fehler
	= -1		Ung ltige Kennung oder Funktion unzul ssig
3 →	**UATTRLEN**	**F**	L nge des Dateinamens bzw. des Bereichs
	= 1 - 3500		bei 8-Bit Komprimat
	= 1 - 1750		bei 7-Bit Komprimat (Mode=CX7)
4 →	**USERATTR**	**XL*n***	Benutzerdaten als bin rer Datenstring.

Bei CX7 werden diese Daten so umgesetzt, dass die Integrit t der FLAMFILE nicht verletzt wird.

Die Benutzerdaten selbst bleiben auch bei Codeumsetzungen im heterogenen Datenaustausch im urspr nglichen Code beim Lesen erhalten.

3.3.22 Funktion FLMPUT

Mit der Funktion FLMPUT wird jeweils ein Originalsatz zum Komprimieren bergeben.

Die Funktion dient zum Erzeugen (Laden) von sequentiellen bzw. indexsequentiellen Komprimatsdateien (OPENMODE=OUTPUT) oder zum Erweitern (OPEN-MODE= INOUT) einer VSAM KSDS FLAMFILE am Dateiende.

Die Satzl nge wird in einem separaten Feld bergeben, sie kann von Satz zu Satz variieren (variable Satzl ngen). Sie wird nicht mit der Angabe der Satzl nge in FLMPHD verglichen.

Bei OPENMODE=OUTPUT und indexsequentieller Organisation wird nicht auf aufsteigende oder doppelte Schl ssel kontrolliert.

Parameter:

1 →	**FLAMID**	**F**	Kennung
2 ←	**RETCO**	**F**	Returncode
	= 0		Kein Fehler
	= -1		Ung ltige Kennung oder Funktion unzul ssig
	= 5		Ung ltiger Schl ssel (doppelt bzw. nicht aufsteigend; nur bei OPEN=INOUT bzw. OPEN=OUTIN)
	= 15		Originalsatz ist gr er als 32763 Bytes
	= 16		Originalsatz ist gr er als Matrix - 4
	= 43		Fehlerabbruch durch Exit
	= x'FFXXXXXX'		DMS-Fehlercode siehe bei FLMOPN
3 →	**RECLEN**	**F**	Satzl nge (Datenl nge) in Bytes ohne Satzl ngenfeld
4 →	**RECORD**	**XL*n***	Originalsatz (Daten)

3.3.23 Funktion FLMPWD

Mit der Funktion FLMPWD wird ein Passwort bergeben. Diese Funktion kann nur einmal aufgerufen werden.

Parameter:

1 →	**FLAMID**	**F**	Kennung
2 ←	**RETCO**	**F**	Returncode
	= 0		Kein Fehler
	= -1		Passwort-Funktion unzul ssig, z.B. f r MODE=CX8, VR8, CX7, bzw. erneuter Aufruf
3 →	**PWDLEN**	**F**	Passwortl nge in Bytes (max. 64)
4 →	**PASSWORD**	**XL*n***	Passwort

3.3.24 Funktion FLMQRY

Mit der Funktion FLMQRY k nnen Parameterwerte erfragt werden.

Sie kann jederzeit nach FLMOPN aufgerufen werden. Die R ckgabewerte sind aber vom Zeitpunkt des Aufrufs abh ngig. So steht z.B. beim Dekomprimieren der Verschl sselungsmode (CRYPTOMODE) erst nach erfolgtem Funktionsaufruf FLMOPF zur Verf gung, Splitt-Parameterwerte erst nach FLMOPD. Allerdings sind alle Werte bekannt, wenn FLMOPN als einzige Open-Funktion (LASTPAR=0) aufgerufen wurde.

Achtung: In Abweichung zu den anderen Funktionsaufrufen wurde das Feld RETCO auf zwei Worte (2 x 4 Byte) erweitert. Das erste Wort gibt wie bisher den Returncode zur Steuerung zur ck, das zweite Wort stellt den INFOCODE dar. In diesem wird der fehlerhafte Parameter zur ckgegeben.

Parameter:

1 → FLAMIDF Kennung

2 ← RETCO,INFCO 2F Returncode, Infocode

= 0,0 Kein Fehler, Infocode=0

ansonsten enth lt der Infocode den Wert des fehlerhaften Parameters

= 91,param unbekannter Parameter

3 → PARAM1 F erster Parameter

4 ← VALUE1 F erster Parameterwert

.
.
.

n → PARAMn F letzter Parameter

n+1← VALUEn F letzter Parameterwert

Hinweis: Es k nnen mehrere Parameter gesetzt werden. Das Ende der Parameterliste ist unbedingt zu markieren. Bei Compilern geschieht das i.d.R. automatisch, in Assembler ist f r die letzte VALUE-Adresse anzugeben : A(X'80000000'+VALUEn).

Folgende Parameterwerte k nnen erfragt werden:

Beschreibung	Parameter	Value
Splitmode	**1**	**0 / 1 / 2** none / serial / parallel
Splitnumber	**2**	**2 - 4**
Cryptomode	**2001**	**0 / 1 / 2** none / FLAM / AES
Secureinfo	**2002**	**0 / 1 / 2 / 3** no / yes / ignore / member

3.3.25 Funktion FLMSET

Funktionserweiterungen der Satzschnitttstelle sind oft mit ˜ nderungen der Parameterlisten verbunden. Um den ˜ nderungsaufwand bei der Programmierung neuer Funktionen so gering als m glich zu halten und die Schnittstellen kompatibel zu lassen, wurde in FLAM V4.1 die Funktion FLMSET implementiert.

Mit der Funktion FLMSET k nnen Parameter bergeben werden.

Diese sind jeweils nach FLMOPN aber vor FLMOPD, bzw. FLMOPF zu bergeben. Ein sp terer Aufruf wird mit Returncode 90 abgewiesen, die Parameter kommen dann nicht zur Wirkung!

Achtung: In Abweichung zu den anderen Funktionsaufrufen wurde das Feld RETCO auf zwei Worte (2 x 4 Byte) erweitert. Das erste Wort gibt wie bisher den Returncode zur Steuerung zur ck, das zweite Wort stellt den INFOCODE dar. In diesem wird der fehlerhafte Parameter zur ckgegeben.

Parameter:

1 → FLAMIDF Kennung

2 ← RETCO,INFCO 2F Returncode, Infocode

= 0,0 Kein Fehler, Infocode=0

ansonsten enth lt der Infocode den Wert des fehlerhaften Parameters

= 90,param falscher Zeitpunkt (z.B. SPLITT nach FLMOPD bei Komprimierung)

= 91,param unbekannter Parameter

= 92,param fehlerhafter Parameterwert

3 → PARAM1 F erster Parameter

4 → VALUE1 F erster Parameterwert

.
.
.

n → PARAMn F letzter Parameter

n+1→ VALUEn	**F**	letzter Parameterwert

Hinweis: Es k nnen mehrere Parameter gesetzt werden. Das Ende der Parameterliste ist unbedingt zu markieren. Bei Compilern geschieht das i.d.R. automatisch, in Assembler ist f r die letzte VALUE-Adresse anzugeben : A(X'80000000'+VALUEn).

Parameter, die VOR der Funktion FLMOPD zu setzen sind:

Beschreibung	Parameter	Value
Splitmode	**1**	**0 / 1 / 2** none / serial / parallel
Splitnumber	**2**	**2 - 4**
Splitsize	**3**	**1 - 4095** Angabe in MegaBytes

Parameter, die VOR der Funktion FLMOPF zu setzen sind:

Beschreibung	Parameter	Value
Cryptomode	**2001**	**0 / 1 / 2** none / FLAM / AES
Secureinfo	**2002**	**0 / 1 / 2 / 3** no / yes / ignore / member

3.3.26 Funktion FLMUPD

Mit der Funktion FLMUPD wird jeweils der zuletzt gelesene Originalsatz aus einer indexsequentiellen FLAMFILE (VSAM-KSDS) ge ndert.

Parameter:

1 →	**FLAMID**	**F**	Kennung
2 ←	**RETCO**	**F**	Returncode
	= 0		Kein Fehler
	= -1		Ung ltige Kennung oder Funktion unzul ssig
	= 5		Kein aktueller Satz vorhanden
	= 15		Originalsatz ist gr er als 32764 Bytes
	= 16		Originalsatz ist gr er als Matrix - 4
	= 43 - 49		Fehlerabbruch durch Exit
	= x'FFXXXXXX'		DMS-Fehlercode
3 →	**RECLEN**	**F**	Satzl nge (Datenl nge) in Bytes ohne Satzl ngenfeld
4 →	**RECORD**	**XLn**	Originalsatz (Daten

3.4 Benutzer Ein-/Ausgabe Schnittstelle

Die Benutzer Ein-/Ausgabe Schnittstelle kann f r das Dienstprogramm FLAM, f r das Unterprogramm FLAMUP und f r die Satzschnittstelle FLAMREC verwendet werden.

Unter FLAM und FLAMUP kann die Eingabedatei (FLAMIN), die Ausgabedatei (FLAMOUT) oder die Komprimatsdatei (FLAMFILE) bearbeitet werden. Die Benutzung dieser Schnittstelle ist durch die Parameter IDEVICE=USER, ODEVICE=USER und DEVICE=USER anzufordern.

An der Satzschnittstelle FLAMREC kann die Benutzer-Ein-/Ausgabe mit dem Parameter DEVICE in der Funktion FLMOPD f r die Komprimatsdatei (FLAMFILE) angefordert werden.

Die entsprechenden Funktionen stellt der Anwender bereit. Dabei sind die Funktionen USROPN und USRCLS obligatorisch. Von den restlichen Funktionen sind nur die bereitzustellen, die f r den jeweiligen Zweck gebraucht werden.

Mit FLAM wird ein Musterprogramm in COBOL und in ASSEMBLER mitgeliefert. In diesem Muster sind f r alle Funktionen Dummys ausprogrammiert.

USROPN ffnen der Datei bzw. Schnittstelle

USRCLS Schlie en der Datei bzw. Schnittstelle

USRGET Einen Satz lesen und bergeben

USRPUT Einen Satz bernehmen und wegschreiben

USRGKY Einen Satz mit Schl ssel lesen und bergeben

USRPOS Weiter positionieren

USRPKY Einen Satz bernehmen und mit Schl ssel wegschreiben

USRDEL Den zuletzt gelesenen Satz l schen

3.4.1 Funktion USROPN

ffen der Schnittstelle f r die im DD-Name angegebenen Datei.

Parameter:

1 ↔	**WORKAREA**	**256F**	Arbeitsbereich ist mit x'00' initialisiert. Dieser Bereich ist der Datei eindeutig zugeordnet. Er kann als Ged chtnis zwischen den Aufrufen benutzt werden.
2 ←	**RETCO**	**F**	Returncode
	=	**0**	Kein Fehler
	=	**-1**	unzul ssige Funktion
	=	**sonst**	siehe Funktion FLMOPN
3 →	**OPENMODE**	**F**	Der Openmode bestimmt die Arbeitsweise
	=	**0**	INPUT (sequentiell lesen) (Datei muss bereits existieren)
	=	**1**	OUTPUT (sequentiell schreiben) (Datei wird neu angelegt oder berschrieben)
	=	**2**	INOUT (mit Schl ssel sowie sequentiell schreiben und lesen) (Datei muss bereits existieren)
	=	**3**	OUTIN (mit Schl ssel sowie sequentiell schreiben und lesen) (Datei wird neu angelegt oder berschrieben)
4 →	**DDNAME**	**CL8**	Symbolischer Dateiname
5 ↔	**DSORG**	**F**	Dateiformat
	=	**0; 8; 16 ...**	sequentiell
	=	**1; 9; 17 ...**	indexsequentiell
	=	**2; 10; 18 ...**	relativ
	=	**3; 11; 19 ...**	Direktzugriff
	=	**5; 13; 21 ...**	Bibliothek
	=	**6; 14; 22 ...**	physikalisch
6 ↔	**RECFORM**	**F**	Satzformat
	=	**0; 8; 16 ...**	**V:** VARIABEL
	=	**1; 9; 17 ...**	**F:** FIX
	=	**2; 10; 18 ...**	**U:** UNDEFINED
	=	**3; 11; 19 ...**	**S:** STREAM
7 ↔	**RECSIZE**	**F**	Satzl nge
	=	**0 bis 32760**	
	RECFORM = V:		Maximale Satzl nge oder 0
	RECFORM = F:		Satzl nge
	RECFORM = U:		Maximale Satzl nge oder 0

	RECFORM = S:		L nge des Texttrenners bzw. L ngenfeldes
8 ↔	**BLKSIZE**	**F**	Blockl nge
	= 0		ungeblockt
	= 1 - 32760		
9 ↔	**KEYDESC STRUCT**		Schl sselbeschreibung
	KEYFLAGS	**F**	Optionen
	= 0		Keine doppelten Schl ssel
	= 1		Doppelte Schl ssel erlaubt
	KEYPARTS	**F**	Anzahl Schl sselteile
	= 0 bis 8		0 = Kein Schl ssel vorhanden
	KEYPOS1	**F**	Erstes Byte des ersten Teilschl ssels
	= 1 bis 32759		Wert kleiner als Satzl nge
	KEYLEN1	**F**	L nge des ersten Teilschl ssels
	= 1 bis 255		
	KEYTYP1	**F**	Datentyp des ersten Teilschl ssels
	= 0		Abdruckbare Zeichen
	= 1		Bin rwert
	.		
	.		
	.		
	KEYPOS8	**F**	Erstes Byte des achten Teilschl ssels
	= 1 bis 32759		Wert kleiner als Satzl nge
	KEYLEN8	**F**	L nge des achten Teilschl ssels
	= 1 bis 255		
	KEYTYP8	**F**	Datentyp des achten Teilschl ssels
	= 0		Abdruckbare Zeichen
	= 1		Bin rwert
10 ↔	**DEVICE**	**F**	Ger tetyp
	= 7; 15; 23 ...		Benutzerger te
11 ↔	**RECDELIM**	**XL*n***	Satztrenner
12 ↔	**PADCHAR**	**XL1**	F llzeichen
13 ↔	**PRCTRL**	**F**	Vorschubsteuerzeichen
	= 0		keine
	= 1		ASA-Steuerzeichen
	= 2		maschinenspezifische Steuerzeichen (MCC)
14 →	**CLOSDISP**	**F**	Art der Close-Bearbeitung
	= 0		REWIND

	= 1		UNLOAD
	= 2		LEAVE
15 →	**ACCESS**	**F**	Zugriffsverfahren
	= 0		logisch (satzweise)
16 ↔	**NAMELEN**	**F**	L nge des Dateinamens bzw. des Bereichs f r den Dateinamen
17 ↔	**FILENAME**	**CL*n***	Dateiname

3.4.2 Funktion USRCLS

Schlie en der Schnittstelle f r eine Datei.

Parameter:

1 ↔	**WORKAREA**	**256F**	Arbeitsbereich
2 ←	**RETCO**	**F**	Returncode
	= 0		Kein Fehler
	= -1		unzul ssige Funktion
	= x'0FXXXXXX'		sonstiger Fehlercode

3.4.3 Funktion USRGET

Satz sequentiell lesen und bergeben.

Parameter:

1 ↔	**WORKAREA**	**256F**	Arbeitsbereich
2 ←	**RETCO**	**F**	Returncode
	= 0		Kein Fehler
	= -1		Funktion unzul ssig
	= 1		Satz wurde verk rzt
	= 2		END-OF-FILE erreicht
	= 3		L cke bei relativer Datei gefunden
	= x'0FXXXXXX'		sonstiger Fehlercode
3 ←	**RECLEN**	**F**	Satzl nge in Bytes des bergebenen Satzes
4 ←	**RECORD**	**XL*n***	Originalsatz (Daten)
5 →	**BUFLEN**	**F**	L nge des verf gbaren Satzpuffers in Bytes

Anmerkung: Bei Returncode 2 (EOF) ist RECLEN auf Null (F'0') zu setzen.

3.4.4 Funktion USRPUT

Satz bernehmen und sequentiell schreiben.

Parameter:

1 ↔	**WORKAREA**	**256F**	Arbeitsbereich
2 ←	**RETCO**	**F**	Returncode
	= 0		Kein Fehler
	= -1		Funktion unzul ssig
	= 1		Satz wurde verk rzt
	= 4		Satz wurde mit F llzeichen (PADCHAR) aufgef llt
	= x'0FXXXXXX'		sonstiger Fehlercode
3 →	**RECLEN**	**F**	Satzl nge in Bytes des bergebenen Satzes
4 →	**RECORD**	**XL*n***	Originalsatz (Daten)

3.4.5 Funktion USRGKY

Satz mit angegebenen Schl ssel lesen und weitergeben. Dabei steht der gesuchte Schl ssel im Satz auf der Schl sselposition laut KEYDESC.

Parameter:

1 ↔	**WORKAREA**	**256F**	Arbeitsbereich
2 ←	**RETCO**	**F**	Returncode
	= 0		Kein Fehler
	= -1		Funktion unzul ssig
	= 1		Satz wurde verk rzt
	= 2		END-OF-FILE erreicht
	= 5		Schl ssel nicht vorhanden
	= x'0FXXXXXX'		sonstiger Fehlercode
3 ←	**RECLEN**	**F**	Satzl nge in Bytes
4 ↔	**RECORD**	**XL*n***	Satz mit Suchbegriff / Satz
5 →	**BUFLEN**	**F**	L nge des verf gbaren Satzpuffers in Bytes

3.4.6 Funktion USRPOS

In Datei positionieren.

Parameter:

1 ↔	**WORKAREA**	**256F**	Arbeitsbereich
2 ←	**RETCO**	**F**	Returncode
	=	**0**	Kein Fehler
	=	**-1**	Funktion unzul ssig
	=	**5**	Unzul ssige Position
	=	**x'0FXXXXXX'**	sonstiger Fehlercode
3 →	**POSITION**	**F**	relative Position
	=	**0**	Keine Positionierung
	=	**- MAXINT**	Dateianfang (-2147483648 bzw. x'80000000')
	=	**+ MAXINT**	Dateiende (+2147483647 bzw. x'7FFFFFFF')
	=	**- n**	n S tze r ckw rts
	=	**+ n**	n S tze vorw rts

Hinweis: Mit dieser Funktion k nnen durch Vorw rtspositionieren in einer relativen Datei L cken erzeugt werden.

3.4.7 Funktion USRPKY

Satz mit angegebenen Schl ssel schreiben.

Parameter:

1 ↔	**WORKAREA**	**256F**	Arbeitsbereich
2 ←	**RETCO**	**F**	Returncode
	=	**0**	Kein Fehler
	=	**-1**	Funktion unzul ssig
	=	**1**	Satz wurde verk rzt
	=	**4**	Satz wurde mit dem F llzeichen (PADCHAR) aufgef llt
	=	**5**	Schl ssel ist ung ltig
	=	**x'0FXXXXXX'**	sonstiger Fehlercode
3 →	**RECLEN**	**F**	Satzl nge in Bytes des bergebenen Satzes

4 → **RECORD** **XL*n*** Originalsatz (Daten)

Hinweis: Der Satz wird normalerweise eingef gt. Nur wenn der Schl ssel des zuletzt gelesenen Satzes mit dem Schl ssel der USRPKY Funktion bereinstimmt, wird der Satz berschrieben (REWRITE). Sonst wird bei gleichem Schl ssel ein weiterer Satz hinzugef gt, sofern doppelte Schl ssel erlaubt sind.

3.4.8 Funktion USRDEL

Den zuletzt gelesenen Satz l schen.

Parameter:

1 ↔	**WORKAREA**	**256F**	Arbeitsbereich
2 ←	**RETCO**	**F**	Returncode
	=	**0**	Kein Fehler
	=	**-1**	Funktion unzul ssig
	=	**5**	Kein aktueller Satz vorhanden
	=	**x'0FXXXXXX'**	sonstiger Fehlercode

3.5 Benutzerausg nge

3.5.1 Eingabe Originaldaten EXK10

In diesem Benutzerausgang werden die zu komprimierenden Originals tze unmittelbar nach dem Lesen von der Eingabedatei zur Verf gung gestellt. Dieser Exit kann in FLAM und FLAMUP benutzt werden. In diesem Benutzerausgang k nnen S tze bernommen, ge ndert, eingef gt und gel scht werden.

Der Exit wird ber den Parameter: EXK10=<name> aktiviert. Er muss dazu in der VSE-Ladebibliothek stehen, die mit LIBDEF zugewiesen wurde.

Name: frei w hlbar (max. 8 Zeichen)

Registerbelegung:

→	**R1:**	Adresse der Parameterliste
→	**R13:**	zeigt auf Sicherstellungsbereich (18 Worte)
→	**R14:**	enth lt die R cksprungadresse
→	**R15:**	enth lt die Aufrufadresse

Parameterliste:

1 →	**FUCO**		**F**	Funktionscode
	=	**0**		erster Aufruf f r die Datei (nach OPEN)
	=	**4**		Satz gelesen und bergeben
	=	**8**		letzter Aufruf f r die Datei (vor CLOSE)
2 ←	**RETCO**		**F**	Returncode
	=	**0**		Satz bernehmen bzw. kein Fehler
	=	**4**		Satz nicht bernehmen
	=	**8**		Satz einf gen
	=	**12**		Ende der Komprimierung einleiten
	=	**16**		Fehler im Exit; abnormales Ende
3 ↔	**RECPTR**		**A**	Satzpointer
4 ↔	**RECLEN**		**F**	Satzl nge (maximal 32760)
5 ↔	**EXWORK**		**256F**	Arbeitsbereich enth lt beim ersten Aufruf den symbolischen Dateinamen der Originaldatei in den ersten 8 Zeichen, der Rest ist mit x’00’ initialisiert. Dieser Bereich kann vom Exit frei verwendet werden. Bei jedem Aufruf wird dieser Arbeitsbereich dem Exit mit altem Inhalt wieder zur Verf gung gestellt.

Hinweise: Soll ein Satz verl ngert oder eingef gt werden, so muss der Speicherbereich daf r im Exit bereitgestellt werden.

Der Returncode 12 ist nur notwendig, wenn die Komprimierung vorzeitig beendet werden soll, ohne dass die Eingabedatei bis zu Ende gelesen wird.

Bei den Funktionscodes 0 und 8 wird kein Satz zur Verf gung gestellt. Es ist jedoch zul ssig, bei Funktionscode 8 einen Satz mit Returncode 8 einzuf gen.

Bei Returncode 8 wird der vom Exit bereitgestellte Satz verarbeitet. Danach wird der Exit mit dem alten Satz der Eingabe erneut aufgerufen.

Funktionscode:		0	4	8
Returncode:	0	x	x	x
	4		x	
	8		x	x
	12		x	
	16	x	x	x

3.5.2 Ausgabe Komprimat EXK20

In diesem Benutzerausgang werden die Komprimatss tze unmittelbar vor dem Schreiben in die Komprimatsdatei zur Verf gung gestellt.

Dieser Exit kann in FLAM, FLAMUP und der Satzschnittstelle FLAMREC benutzt werden. In diesem Benutzerausgang k nnen S tze bernommen, ge ndert, eingef gt und gel scht werden.

Der Exit wird ber den Parameter: EXK20=<name> aktiviert. Er muss dazu in der VSE-Ladebibliothek stehen, die mit LIBDEF zugewiesen wurde.

Name: frei w hlbar (max. 8 Zeichen)

Registerbelegung:

→	**R1:**	Adresse der Parameterliste
→	**R13:**	zeigt auf Sicherstellungsbereich (18 Worte)
→	**R14:**	enth lt die R cksprungadresse
→	**R15:**	enth lt die Aufrufadresse

Parameterliste:

1 →	**FUCO**	**F**	Funktionscode
	= 0		erster Aufruf f r die Datei (nach OPEN)
	= 4		Satz bergeben
	= 8		letzter Aufruf f r die Datei (vor CLOSE)
2 ←	**RETCO**	**F**	Returncode
	= 0		Satz bernehmen bzw. kein Fehler
	= 4		Satz nicht bernehmen
	= 8		Satz einf gen
	= 12		Ende der Komprimierung einleiten
	= 16		Fehler im Exit; abnormales Ende
3 ↔	**RECPTR**	**A**	Satzpointer
4 ↔	**RECLEN**	**F**	Satzl nge (maximal 32760)
5 ↔	**EXWORK**	**256F**	Arbeitsbereich enth lt beim ersten Aufruf den symbolischen Dateinamen der Originaldatei in den ersten 8 Zeichen, der Rest ist mit x'00' initialisiert. Dieser Bereich kann vom EXIT frei verwendet werden. Bei jedem Aufruf wird dieser Arbeitsbereich dem Exit mit altem Inhalt wieder zur Verf gung gestellt.

Hinweise: Soll ein Satz verl ngert oder eingef gt werden, so muss der Speicherbereich daf r im Exit bereitgestellt werden.

Der Returncode 12 ist nur notwendig, wenn die Komprimierung vorzeitig beendet werden soll, ohne dass die Eingabedatei bis zu Ende gelesen wird.

Bei den Funktionscodes 0 und 8 wird kein Satz zur Verf gung gestellt. Es ist jedoch zul ssig, bei Funktionscode 8 einen Satz mit Returncode 8 einzuf gen.

Bei Returncode 8 wird der vom Exit bereitgestellte Satz geschrieben. Danach wird der Exit mit dem alten Komprimatssatz erneut aufgerufen.

Funktionscode:		0	4	8
Returncode:	0	x	x	x
	4		x	
	8		x	x
	12		x	
	16	x	x	x

3.5.3 Ausgabe Originaldaten EXD10

In diesem Benutzerausgang werden die dekomprimierten Originals tze unmittelbar vor dem Schreiben in die Ausgabedatei zur Verf gung gestellt. Dieser Exit kann in FLAM und FLAMUP benutzt werden. Hier k nnen S tze bernommen, ge ndert, eingef gt und gel scht werden.

Der Exit wird ber den Parameter: EXD10=<name> aktiviert. Er muss dazu in der VSE-Ladebibliothek stehen, die mit LIBDEF zugewiesen wurde.

Name: frei w hlbar (max. 8 Zeichen)

Registerbelegung:

→	**R1:**	Adresse der Parameterliste
→	**R13:**	zeigt auf Sicherstellungsbereich (18 Worte)
→	**R14:**	enth lt die R cksprungadresse
→	**R15:**	enth lt die Aufrufadresse

Parameterliste:

1 →	**FUCO**		**F**	Funktionscode
	=	**0**		erster Aufruf f r die Datei (nach OPEN)
	=	**4**		Satz bergeben
	=	**8**		letzter Aufruf f r die Datei (vor CLOSE)
2 ←	**RETCO**		**F**	Returncode
	=	**0**		Satz bernehmen bzw. kein Fehler
	=	**4**		Satz nicht bernehmen
	=	**8**		Satz einf gen
	=	**12**		Ende der Dekomprimierung einleiten
	=	**16**		Fehler im Exit; abnormales Ende
3 ↔	**RECPTR**		**A**	Satzpointer
4 ↔	**RECLEN**		**F**	Satzl nge (maximal 32760)
5 ↔	**EXWORK**		**256F**	Arbeitsbereich enth lt beim ersten Aufruf den symbolischen Dateinamen der Originaldatei in den ersten 8 Zeichen, der Rest ist mit x'00' initialisiert. Dieser Bereich kann vom EXIT frei verwendet werden. Bei jedem Aufruf wird dieser Arbeitsbereich dem Exit mit altem Inhalt wieder zur Verf gung gestellt.

Hinweise: Soll ein Satz verl ngert oder eingef gt werden, so muss der Speicherbereich daf r im Exit bereitgestellt werden.

Der Returncode 12 ist nur notwendig, wenn die Dekomprimierung vorzeitig beendet werden soll, ohne dass die Komprimatsdatei bis zum Ende gelesen wird.

Bei den Funktionscodes 0 und 8 wird kein Satz zur Verf gung gestellt. Es ist jedoch zul ssig, bei Funktionscode 8 einen Satz mit Returncode 8 einzuf gen.

Bei Returncode 8 wird der vom Exit bereitgestellte Satz geschrieben. Danach wird der Exit mit dem alten Satz erneut aufgerufen.

Eine ˜ nderung der Satzl nge wird nur ber cksichtigt, wenn die Ausgabedatei mit RECFORM=V definiert ist.

3.5.4 Eingabe Komprimat EXD20

In diesem Benutzerausgang werden die Komprimatss tze unmittelbar nach dem Lesen aus der Komprimatsdatei zur Verf gung gestellt. Dieser Exit kann in FLAM, FLAMUP und der Satzschnittstelle FLAMREC benutzt werden. In diesem Benutzerausgang k nnen S tze bernommen, ge ndert und gel scht werden.

Der Exit wird ber den Parameter EXD20=<name> aktiviert. Er muss dazu in der VSE-Ladebibliothek stehen, die mit LIBDEF zugewiesen wurde.

Name: frei w hlbar (max. 8 Zeichen)

Registerbelegung:

→	**R1:**	Adresse der Parameterliste
→	**R13:**	zeigt auf Sicherstellungsbereich (18 Worte)
→	**R14:**	enth lt die R cksprungadresse
→	**R15:**	enth lt die Aufrufadresse

Parameterliste:

1 →	**FUCO**		**F**	Funktionscode
	=	**0**		erster Aufruf f r die Datei (nach OPEN)
	=	**4**		Satz bergeben
	=	**8**		letzter Aufruf f r die Datei (vor CLOSE)
2 ←	**RETCO**		**F**	Returncode
	=	**0**		Satz bernehmen bzw. kein Fehler
	=	**4**		Satz nicht bernehmen
	=	**8**		Satz einf gen
	=	**12**		Ende der Dekomprimierung einleiten
	=	**16**		Fehler im Exit; abnormales Ende
3 ↔	**RECPTR**		**A**	Satzpointer
4 ↔	**RECLEN**		**F**	Satzl nge (maximal 32760)
5 ↔	**EXWORK**		**256F**	Arbeitsbereich enth lt beim ersten Aufruf den symbolischen Dateinamen der Originaldatei in den ersten 8 Zeichen, der Rest ist mit x'00' initialisiert. Dieser Bereich kann vom EXIT frei verwendet werden. Bei jedem Aufruf wird dieser Arbeitsbereich dem Exit mit altem Inhalt wieder zur Verf gung gestellt.

Hinweise: Soll ein Satz verl ngert oder eingef gt werden, so muss der Speicherbereich daf r im Exit bereitgestellt werden.

Der Returncode 12 ist nur notwendig, wenn die Dekomprimierung vorzeitig beendet werden soll, ohne dass die Komprimatsdatei bis zum Ende gelesen wird.

Wegen der notwendigen Synchronisation mit dem Aufbau einer Matrix, ist dieser Returncode nur bedingt einsetzbar.

Bei den Funktionscodes 0 und 8 wird kein Satz zur Verf gung gestellt.

Funktionscode:		0	4	8
Returncode:	0	x	x	x
	4		,	
	8		,	x
	12		(x)	
	16	x	x	x

3.5.5 Schl sselverwaltung KMEXIT

Dieser Benutzerausgang dient zum Anschluss an ein Schl sselverwaltungssystem (Key Management).

Die Aufgabe dieser Benutzerroutine ist es, zur Ver- / Entschl sselung einer FLAMFILE einen Schl ssel zur Verf gung zu stellen.

Er kann in FLAM und FLAMUP benutzt werden.

Der Exit wird ber den Parameter KMEXIT=name aktiviert. Er wird dann f r jede FLAMFILE aufgerufen.

Beim Aufruf durch FLAM werden die Angaben des Parameters KMPARM (max. 256 Bytes) zur Verf gung gestellt.

Der Exit kann bei der Verschl sselung einen Datenstring von max. 512 Bytes zur ckgeben, der in der FLAMFILE gespeichert wird (als Userheader, vgl. Funktion FLMPUH). Zur Entschl sselung werden diese Daten wieder von FLAM an den Exit bergeben.
Die ersten 50 Zeichen des Datenstrings werden bei der Dekomprimierung/Entschl sselung als Kommentar im Protokoll angezeigt (FLM0482 OLD COMMENT: ...).

Name: frei w hlbar (max. 8 Zeichen)

Registerbelegung:

→	**R1:**	Adresse der Parameterliste
→	**R13:**	zeigt auf Sicherstellungsbereich (18 Worte)
→	**R14:**	enth lt die R cksprungadresse
→	**R15:**	enth lt die Aufrufadresse

Parameterliste:

1 →	**FUCO**	**F**	Funktionscode
	= 0		Entschl sselung
	= 1		Verschl sselung
2 ←	**RETCO**	**F**	Returncode
	= 0		kein Fehler
	= sonst		Fehler
3 →	**PARMLEN**	**F**	L nge Parameter (max. 256)
4 →	**PARAM**	**XLn**	Parameter (in L nge PARMLEN)

5 ↔ DATALEN F Datenl nge

Entschl sselung:

→ L nge Daten

Verschl sselung:
→ Gr   e Feld DATA (512)
← L nge Daten (max. 512)

6 ↔ DATA XLn Daten (in L nge DATALEN)

7 ↔ CKYLEN F Schl ssell nge
→ Gr   e Schl sselpuffer (Feld CRYPTOKEY) (64)
← L nge Schl ssel (max. 64)

8 ← CRYPTOKEY XLn Schl ssel (in L nge CKYLEN)

9 ↔ MSGLEN F Nachrichtenl nge
→ Gr   e Nachrichtenpuffer (Feld MESSAGE) (128)
← L nge Nachricht (max. 128)

10 ← MESSAGE CLn Nachricht (in L nge MSGLEN)

Wird eine Nachrichtenl nge > 0 zur ckgegeben, wird die Meldung FLM0445 … im Protokoll ausgegeben.

Die Daten DATA werden unver ndert im Userheader der FLAMFILE gespeichert. Wird ein spezieller Schutz gew nscht, ist er vom Exit selbst zu realisieren.

Bei Verwendung dieses Exits werden die FLAM Parameter (z.B. der Kommandozeile) COMMENT und CRYPTOKEY   berschrieben.

Der   bergebene Schl ssel wird NICHT protokolliert.

Der Exit wird pro FLAMFILE nur ein Mal aufgerufen. D.h. werden mehrere Dateien in eine Sammel-FLAMFILE komprimiert (C,FLAMIN=user.*), erfolgt der Aufruf nur ein Mal zu Beginn. Werden aber mehrere FLAMFILEs gelesen (D,FLAMFILE=user.*.aes), wird nach jedem   ffnen einer FLAMFILE der Exit aufgerufen. Konkatinierte FLAMFILEs gelten als eine Datei!

Hinweis: Ein funktionsf higes Beispiel ist in der ausgelieferten Bibliothek FLAM. LIB(KMXSAMPL.A) enthalten.

FLAM (VSE)

Benutzerhandbuch

Kapitel 4:

Arbeitsweise

Inhalt

4. Arbeitsweise

W hrend die vorangegangenen Kapitel beschreiben, wo Komprimierung sinnvoll einzusetzen ist, welche Funktionen von FLAM dazu angeboten werden und in der jeweiligen Umgebung genutzt werden k nnen, erkl rt dieses Kapitel die interne Arbeitsweise f r den effizienten Einsatz dieses Produktes.

Es wird unterschieden zwischen einem Dienstprogramm zur Verarbeitung ganzer Dateien, das als Haupt- oder Unterprogramm aufgerufen werden kann, und Schnittstellen zur satzweisen Verarbeitung von Daten, die von einem Anwenderprogramm bergeben bzw. bernommen werden k nnen.

Dienstprogramm

Das Dienstprogramm kann direkt unter dem Betriebssystem durch ein Kommando gestartet werden. Dabei wird ber Parameter die Art der Verarbeitung gesteuert. Je nach Betriebssystem k nnen die Parameter direkt im Kommando mitgegeben oder in einem Dialog am Bildschirm eingegeben werden.

Zus tzlich k nnen Parameter auch aus einer Datei gelesen werden. Die Dateien werden ber die Kommandosprache des Betriebssystems oder ber Parameter zugeordnet und spezifiziert.

Unterprogramm

Das Unterprogramm bietet die gleiche Funktionalit t wie das Hauptprogramm. Es kann jedoch von einem Anwenderprogramm aus aufgerufen werden. Bei diesem Aufruf k nnen Parameter mitgegeben werden.

Satzschnittstelle

ber die Satzschnittstelle k nnen Daten von einem Anwenderprogramm satzweise komprimiert bzw. dekomprimiert werden. FLAM verwaltet die Komprimatsdatei unterhalb dieser Schnittstelle. Von einem Anwenderprogramm k nnen mehrere Komprimatsdateien gleichzeitig verarbeitet werden. F r das Anwenderprogramm bildet die Satzschnittstelle eine equivalente Schnittstelle zum Dateizugriff des Betriebssystems mit dem Unterschied, dass die Daten komprimiert gespeichert werden und dass die Satzschnittstelle auf allen Betriebssystemen gleich ist.

Benutzer Ein-/Ausgabe

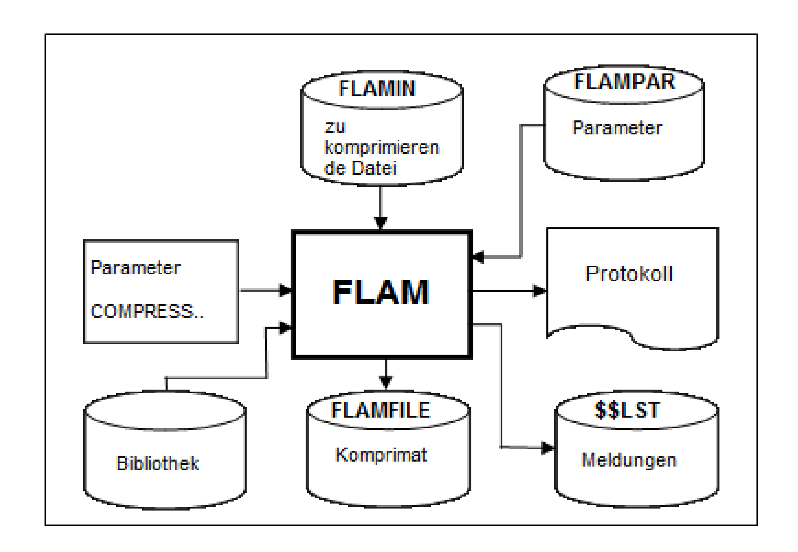
Die Benutzerschnittstelle f r Ein-/Ausgabe erm glicht den Austausch mitgelieferter Dateizugriffsfunktionen durch Funktionen, die vom Benutzer bereitgestellt werden. ber diese Schnittstelle k nnen sowohl Originaldateien im Dienstprogramm als auch die Komprimatsdatei im Dienstprogramm und unter der Satzschnittstelle bearbeitet werden.

Benutzerausg nge

 ber Benutzerausg nge k nnen Vor- und Nachbearbeitungen von S tzen durchgef hrt werden. Es k nnen Originals tze im Dienstprogramm vor der Komprimierung und nach der Dekomprimierung bearbeitet werden. Komprimatss tze k nnen im Dienstprogramm und unter der Satzschnittstelle bearbeitet werden. Diese Benutzerausg nge dienen beispielsweise zur Verschl sselung von Komprimaten oder zur selektiven Verarbeitung von Originaldaten.

4.1 Verarbeiten von Dateien mit dem Dienstprogramm

4.1.1 Komprimieren



Datenfluss bei Komprimierung

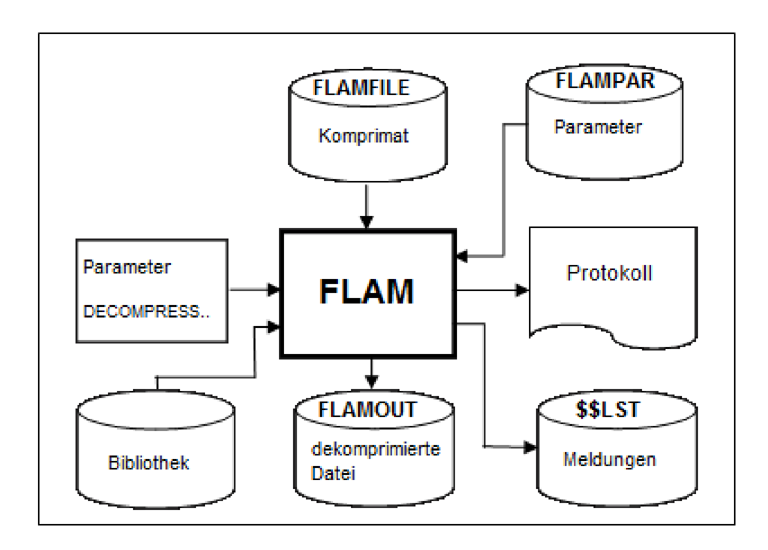
FLAM liest die unkomprimierten Datens tze von der Originaldatei, komprimiert sie und schreibt sie in die Komprimatsdatei.

FLAM ben tigt Angaben ber die Art der Komprimierung, die zu komprimierende Datei und die Komprimatsdatei.

Die so erstellte Komprimatsdatei kann mit dem Dienstprogramm FLAM, mit dem Unterprogramm FLAMUP oder mit der Satzschnittstelle FLAMREC dekomprimiert werden.

Wahlweise ist die Ausgabe eines Protokolls m glich.

4.1.2 Dekomprimieren



Datenfluss bei Dekomprimierung

FLAM liest die komprimierten Datens tze von der Komprimatsdatei, dekomprimiert sie und schreibt sie in die Ausgabedatei.

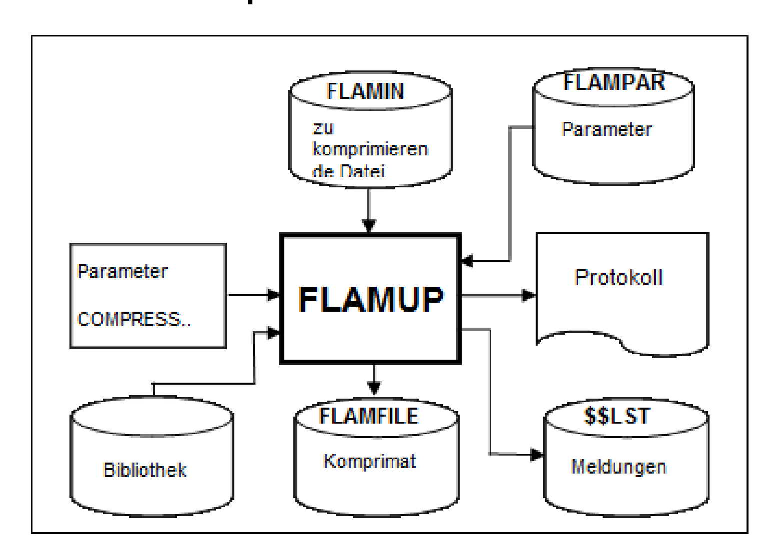
Sind die Dateiattribute der Originaldatei nicht bekannt (kein Fileheader), so muss der Anwender die Dateiattribute per Parameter oder durch Kommandos vorgeben. FLAM erzeugt sonst eine sequentielle Datei mit variabler Satzl nge.

FLAM ben tigt f r die Dekomprimierung einer Datei die Zuweisung der Komprimats- und der Ausgabedatei.

Wahlweise ist die Ausgabe eines Protokolls m glich.

4.2 Verarbeiten von Dateien mit dem Unterprogramm

4.2.1 Komprimieren



Datenfluss bei Komprimierung

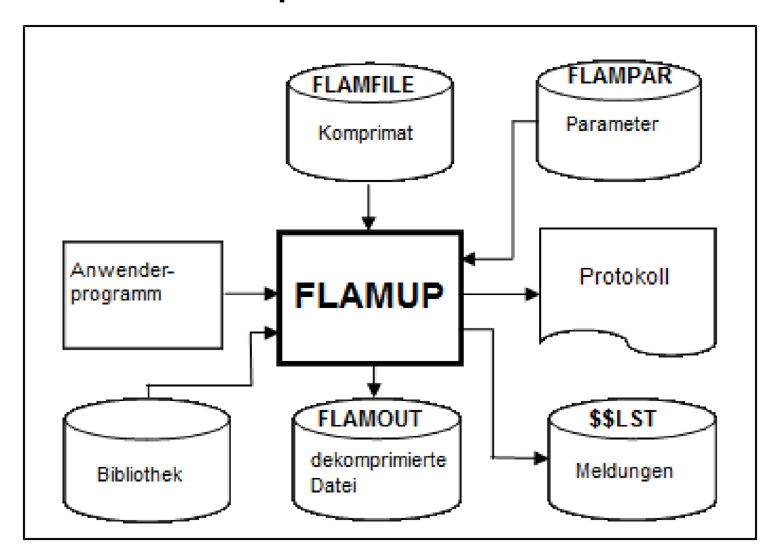
FLAMUP liest, wie FLAM, die unkomprimierten Datens tze von der Originaldatei, komprimiert sie und schreibt sie in die Komprimatsdatei.

FLAMUP ben tigt f r die Komprimierung, wie FLAM, die Zuordnung der Original- und der Komprimatsdatei.

Parameter k nnen beim Aufruf bzw. ber eine Parameterdatei angegeben werden.

Die Ausgabe eines Protokolls ist wahlweise m glich.

4.2.2 Dekomprimieren



Datenfluss bei Dekomprimierung

FLAMUP liest, wie FLAM, die komprimierten Datens tze von der Komprimatsdatei, dekomprimiert sie und schreibt sie in eine Ausgabedatei. Die Ausgabedatei ist wahlweise mit den gleichen Dateiattributen der Originaldatei oder nach den Vorgaben des Anwenders einzurichten.

FLAMUP ben tigt f r die Dekomprimierung einer Datei Angaben ber die dekomprimierte Ausgabedatei und die Komprimatsdatei, analog zum Dekomprimieren mit FLAM.

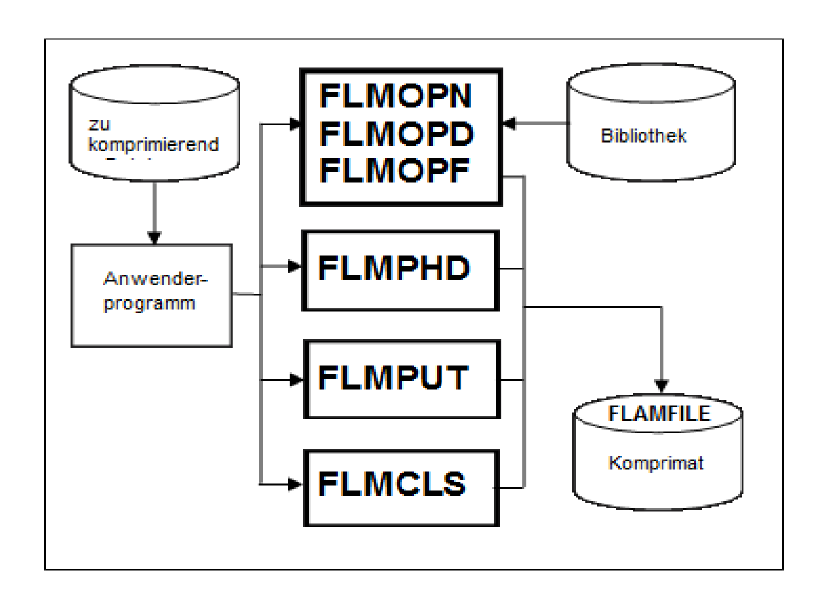
Parameter k nnen beim Aufruf bergeben bzw. aus einer Parameterdatei gelesen werden.

Wahlweise ist die Ausgabe eines Protokolls m glich.

4.3 Verarbeiten von S tzen mit der Satzschnittstelle

ber die Satzschnittstelle k nnen Daten von einem Anwenderprogramm satzweise komprimiert bzw. dekomprimiert werden. FLAM verwaltet die Komprimatsdatei unterhalb dieser Schnittstelle. Von einem Anwenderprogramm k nnen mehrere Komprimatsdateien gleichzeitig verarbeitet werden. F r das Anwenderprogramm bildet die Satzschnittstelle eine equivalente Schnittstelle zum Dateizugriff des Betriebssystems mit dem Unterschied, dass die Daten komprimiert gespeichert werden und dass die Satzschnittstelle auf allen Betriebssystemen gleich ist.

4.3.1 Komprimieren



Datenfluss bei Komprimierung

ber die Satzschnittstelle gibt das Anwendungsprogramm die S tze zum Komprimieren direkt an FLAM weiter. FLAM sammelt die S tze, bis die maximale Anzahl von S tzen (MAXRECORDS) in einem Block erreicht oder der zur Verf gung stehende Puffer (MAXBUFFER) gef llt ist. Die Daten werden komprimiert und die komprimierten S tze in eine Datei geschrieben. Danach k nnen die Datens tze f r den n chsten Block bergeben und komprimiert werden. F r den Anwender bleibt die Blockbildung unsichtbar. Er bergibt nur seine

Datens tze, FLAM bildet die Bl cke und f hrt die Komprimierung durch.

Die bergabe der Datens tze vom Anwenderprogramm an der Satzschnittstelle wird ber verschiedene Funktionen (FLMOPN, ... ,FLMCLS) gesteuert.

Reihenfolge der Funktionsaufrufe:

1. FLMOPN

ffnen der Satzschnittstelle zum Schreiben, ggf. folgen noch FLMOPD und FLMOPF zum Einstellen bestimmter Parameter.

2. FLMPHD

bergeben der Fileheader-Informationen (wahlfrei).

3. FLMPUT

bergabe eines Originalsatzes, mit Wiederholung bis alle S tze an FLAM bergeben wurden.

4. FLMCLS

Schlie en der Satzschnittstelle und gegebenenfalls die Entgegennahme der Statistikdaten.

Die Ausgabe eines Protokolls und die bergabe von Parametern aus einer Datei ist nicht vorgesehen.

4.3.2 Dekomprimieren

FLMOPN
FLMOPD
FLMOPF
Bibliothek
Anwender-
programm
FLMGHD
FLMGET
FLMLOC
FLAMFILE
Komprimat
dekomprimierte
Datei
FLMCLS

Datenfluss bei Dekomprimierung

Die Satzschnittstelle bergibt dem Anwenderprogramm die dekomprimierten S tze direkt von FLAM. Die S tze k nnen sequentiell bzw. ber Satzschl ssel gelesen werden. FLAM liest die Komprimatss tze blockweise und dekomprimiert die Bl cke automatisch. Das Anwenderprogramm nimmt von dieser blockweisen Verarbeitung keine Kenntnis. Das Ende der Komprimatsdatei bzw. das Ende einer Originaldatei in einem Sammelkomprimat wird ber einen Returncode gemeldet.

Die bernahme der Datens tze durch das Anwenderprogramm an der Satzschnittstelle wird durch verschiedene Funktionen (FLMOPN, ... ,FLMCLS) gesteuert.

Reihenfolge der Funktionsaufrufe:

1. FLMOPN

 ffnen der Satzschnittstelle zum Lesen, gegebenenfalls folgen FLMOPD und FLMOPF zum Einstellen bzw. Ermitteln bestimmter Parameter.

2. FLMGHD

 bernehmen der Fileheader-Informationen (wahlfrei). Kann gegebenenfalls wiederholt werden, wenn in einem Sammelkomprimat eine neue Datei beginnt.

3. FLMGET

 bernehmen eines dekomprimierten Originalsatzes. Kann solange wiederholt werden, bis alle S tze von FLAM bernommen oder die Schnittstelle mit FLMCLS geschlossen wird.

4. FLMCLS

Schlie en der Satzschnittstelle und gegebenenfalls Entgegennahme der Statistikdaten.

Die Ausgabe eines Protokolls und die bergabe von Parametern aus einer Datei ist nicht vorgesehen.

4.4 Benutzer Ein-/Ausgabe

Mit Hilfe der Benutzer-Ein-/Ausgabe-Schnittstelle k nnen die in FLAM enthaltenen Dateizugriffsfunktionen durch eigene Routinen des Anwenders ersetzt werden.

Diese Routinen werden im Dienstprogramm f r die Bearbeitung der Originaldateien und die Komprimatsdatei eingesetzt. Unter der Satzschnittstelle kann nur die Komprimatsdatei bearbeitet werden.

Die Verwendung der benutzerspezifischen Ein-/Ausgabe wird f r jede Datei ber den Parameter DEVICE=USER bzw. IDEVICE, ODEVICE getrennt eingestellt. Dazu m ssen die Routinen zur benutzerspezifischen Ein-/Ausgabe zuvor in das Dienstprogramm oder die Satzschnittstelle eingebunden werden.

Es m ssen Routinen zum ffnen und Schlie en (USROPN, USRCLS) der Dateien und zum sequentiellen Schreiben und Lesen (USRPUT, USRGET) bereitgestellt werden. Das gilt gegebenenfalls auch zum Schreiben und Lesen ber Schl ssel (USRPKY, USRGKY) bzw. zum L schen und Positionieren (USRDEL, USRPOS).

Arbeitsweise:

1. USROPN:

F r jede zugeordnete Datei wird diese Funktion als erste genau einmal aufgerufen. Es wird ein Arbeitsbereich von 1024 Bytes als dateispezifisches Ged chtnis zur Verf gung gestellt. Dieser Bereich wird bei allen nachfolgenden Aufrufen bis zum USRCLS unver ndert weitergegeben.

Die Zuordnung der Datei erfolgt ber den symbolischen Dateinamen. Im Parameter OPENMODE wird die Art des gew nschten Zugriffs: INPUT, OUTPUT, INOUT, OUTIN spezifiziert. In den Parametern RECFORM, RECSIZE, BLKSIZE usw. , werden die Dateiattribute spezifiziert, die gegebenenfalls an die Gegebenheiten der Datei angepasst werden k nnen.

ber fest definierte und frei vergebbare Returncodes k nnen der erfolgreiche Abschluss der Funktion, bzw. spezielle Zust nde und Fehler gemeldet werden. Der Returncode wird von FLAM ausgewertet und im Falle eines Fehlers an die oberen Schichten weitergeleitet.

2. USRCLS:

Mit dieser Funktion wird das Schlie en der Datei veranlasst. Der Arbeitsbereich f r diese Datei wird von FLAM nach R ckgabe der Kontrolle wieder freigegeben.

3. USRGET:

Mit dieser Funktion wird der n chste Satz angefordert. Es d rfen maximal so viele Zeichen bergeben werden wie im Parameter BUFLEN angegeben sind. Muss der Satz deshalb verk rzt werden, ist das im Returncode zu melden. Wird das Dateiende erreicht, ist das ebenfalls im Returncode zur ckzumelden. F r jeden gelesenen Satz ist die Satzl nge zur ckzugeben (auch bei fixem Satzformat).

4. USRPUT:

Mit dieser Funktion wird ein Satz zum Schreiben bergeben. Kann der Satz nicht in der angegebenen L nge geschrieben werden, ist die Verk rzung im Returncode zu melden. Oder der Satz muss mit dem beim USROPN angegebenen F llzeichen (PADCHAR) aufgef llt und der entsprechende Returncode zur ckgemeldet werden.

5. USRPOS:

Mit dieser Funktion wird die aktuelle Schreib-/Leseposition ge ndert. Es sind relative Positionierungen um n-S tze vorw rts bzw. r ckw rts und absolute Positionierungen an den Dateianfang bzw. das Ende m glich.

6. USRGKY:

Mit dieser Funktion wird ein Satz mit einem bestimmten Schl ssel gelesen. Der gew nschte Schl ssel steht im Satzbereich an der Position und mit der L nge wie es in der Schl sselbeschreibung (KEYDESC) beim USROPN festgelegt wurde. Das Lesen ber Schl ssel legt auch die Position f r nachfolgende sequentielle Lesefunktionen (USRGET) fest. Wird ein Satz nicht gefunden, muss das mit einem entsprechenden Returncode zur ckgemeldet werden. Mit USRGET kann dann der Satz mit dem n chst gr eren Schl ssel gelesen werden.

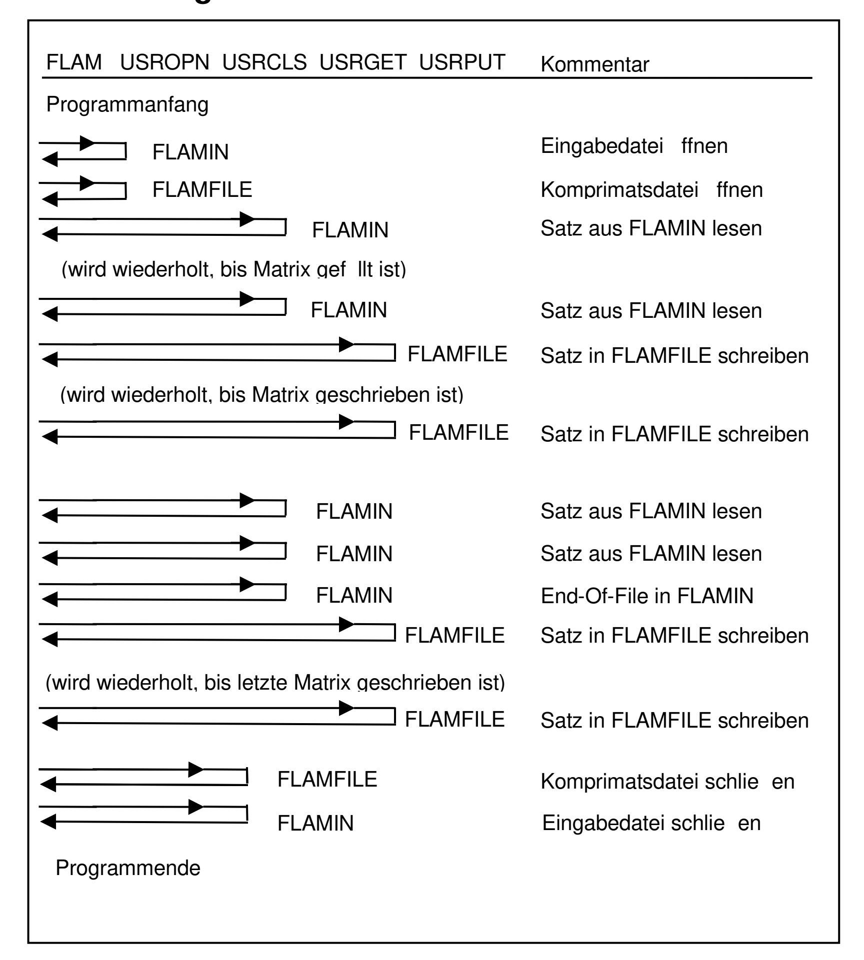
7. USRPKY:

Mit dieser Funktion wird ein Satz mit dem angegebenen Schl ssel ersetzt oder eingef gt. Hat der Satz den gleichen Schl ssel wie der zuletzt gelesene Satz, so wird er durch den aktuellen ersetzt. Im anderen Fall wird der Satz eingef gt. Ist dies nicht m glich, weil z.B. keine doppelten Schl ssel erlaubt sind, so ist dies mit einem entsprechenden Returncode zur ckzumelden. Das Schreiben ber Schl ssel legt auch die Position f r nachfolgende sequentielle Schreibfunktionen (USRPUT) fest.

8. USRDEL:

Mit dieser Funktion wird der zuletzt gelesene Satz gel scht.

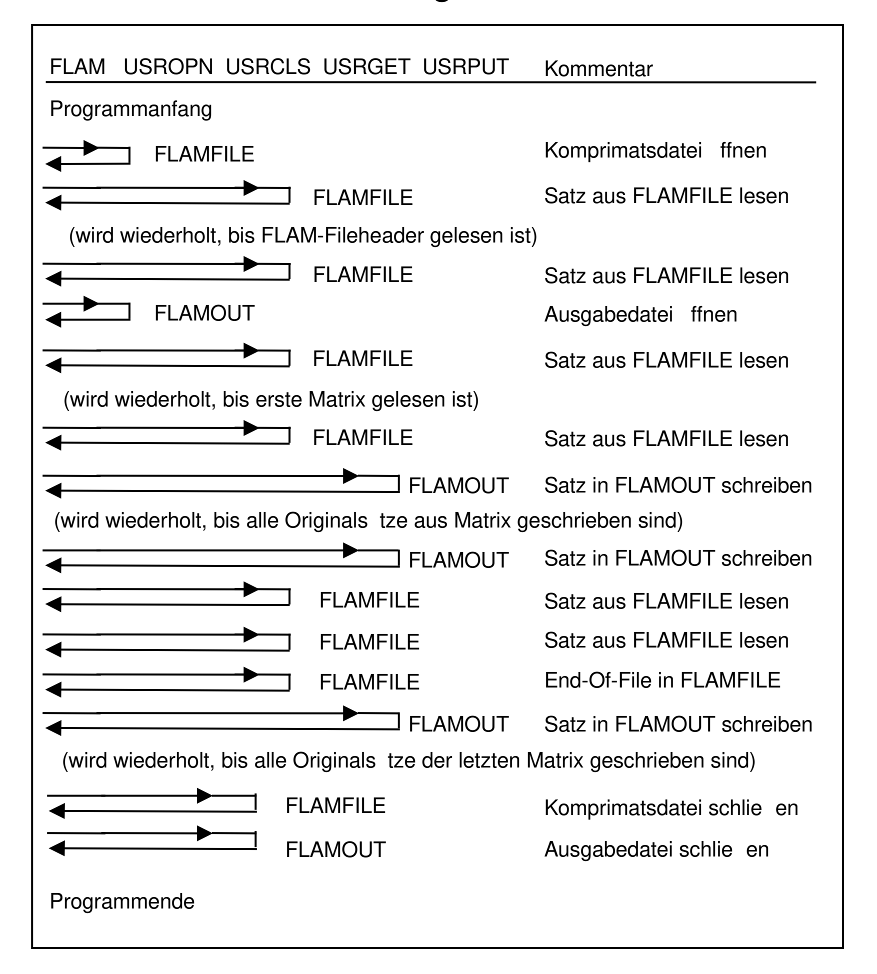
Komprimierung mit USER-IO in schematischer Darstellung:



Parameter f r FLAM oder FLAMUP:

COMPRESS, IDEVICE = USER, DEVICE=USER

Dekomprimierung mit USER-IO in schematischer Darstellung:



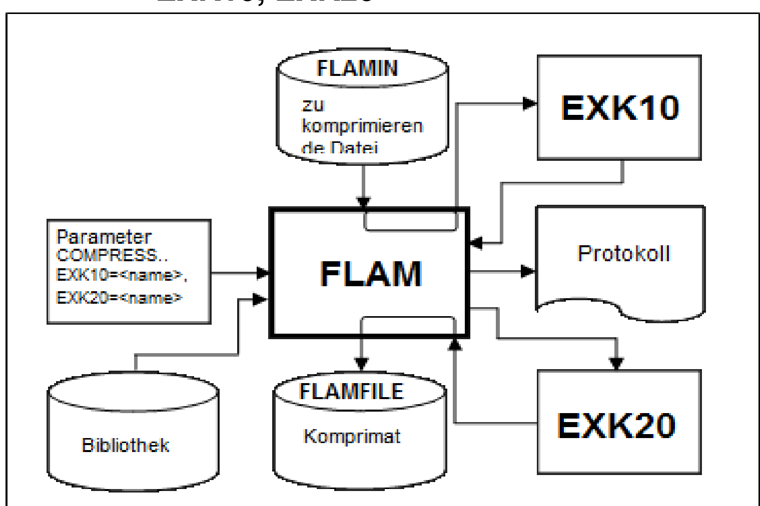
Parameter f r FLAM oder FLAMUP:

DECOMPRESS, ODEVICE = USER, DEVICE = USER

4.5 Benutzerausg nge

4.5.1 Dienstprogramm

4.5.1.1 Komprimieren mit Benutzerausg ngen EXK10, EXK20



Datenfluss bei Komprimierung mit Benutzerausg ngen

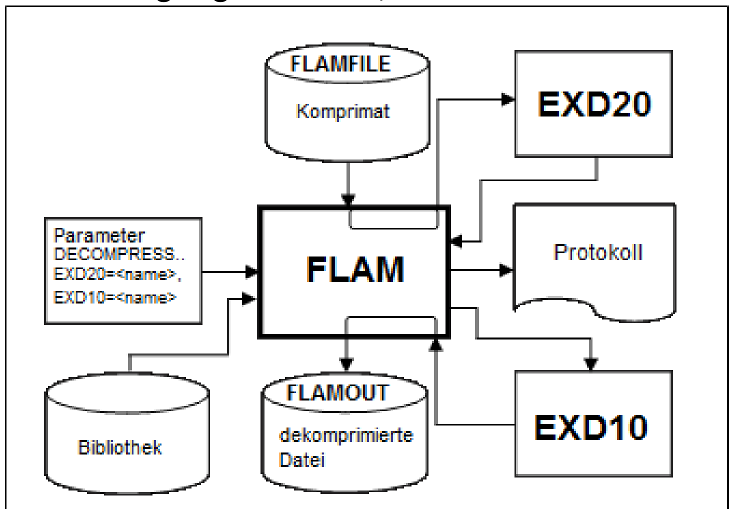
Bei der Komprimierung k nnen zus tzlich Routinen zur Vorbearbeitung der Originals tze und zur Nachbereitung der Komprimatss tze aufgerufen werden.

Die Vorbearbeitung der Originals tze kann z.B. eine Selektion von S tzen oder Feldern sein.

Die Nachbearbeitung der Komprimatss tze kann z.B. eine Verschl sselung des Komprimats sein.

In vielen F llen k nnen anstelle einer aufwendigeren Implementierung mit Hilfe der Satzschnittstelle, die Verarbeitungen satzweise mit dem Benutzerausgang EXK10 durchgef hrt werden.

4.5.1.2 Dekomprimieren mit Benutzeraus-g ngen EXD10, EXD20



Datenfluss bei Dekomprimierung mit Benutzerausg ngen

Bei der Dekomprimierung k nnen zus tzlich Routinen zur Vorbearbeitung der Komprimatss tze und zur Nachbereitung der Originals tze aufgerufen werden.

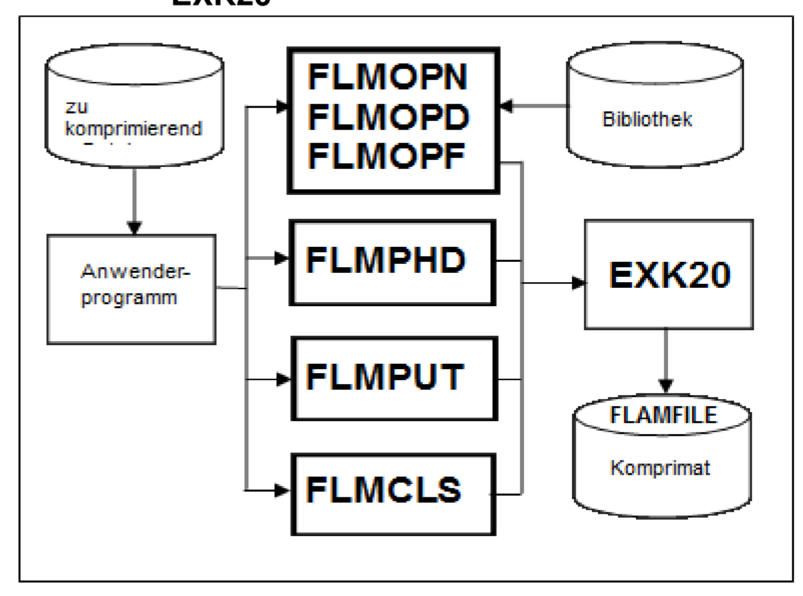
Die Vorbearbeitung der Komprimatss tze kann z.B. eine Entschl sselung des Komprimats sein.

Die Nachbearbeitung der Originals tze kann z.B. eine Selektion von S tzen oder Feldern sein.

In vielen F llen k nnen anstelle einer aufwendigeren Implementierung mit Hilfe der Satzschnittstelle, die Verarbeitungen satzweise mit dem Benutzerausgang EXD10 durchgef hrt werden.

4.5.2 Satzschnittstelle

4.5.2.1 Komprimieren mit Benutzerausgang EXK20

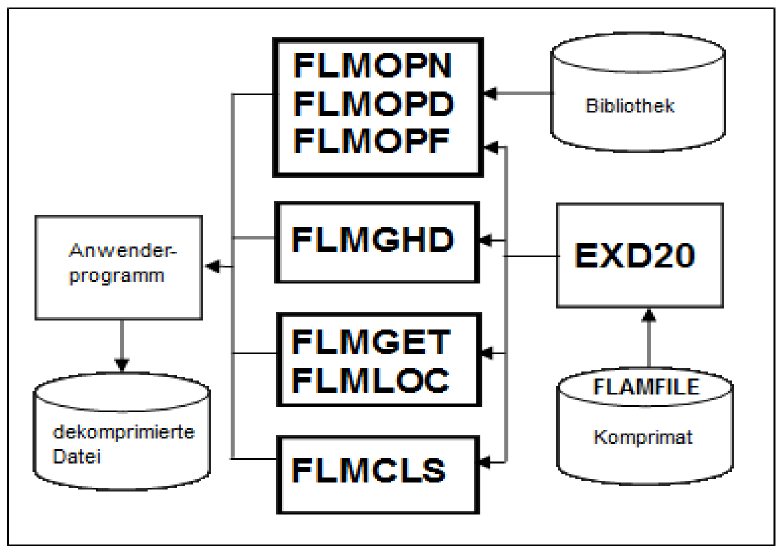


Datenfluss bei Komprimierung mit Benutzerausgang

Der Benutzerausgang f r Komprimatss tze kann auch unterhalb der Satzschnittstelle benutzt werden.

An der bergabe der Originals tze ndert sich dadurch nichts.

4.5.2.2 Dekomprimieren mit Benutzerausgang EXD20



Datenfluss bei Dekomprimierung mit Benutzerausgang

Der Benutzerausgang f r Komprimatss tze kann auch unterhalb der Satzschnittstelle benutzt werden.

An der bernahme der Originals tze ndert sich dadurch nichts.

4.6 Die FLAMFILE

4.6.1 Allgemeine Beschreibung

Unabh ngig von der Komprimierungstechnik des Frankenstein-Limes-Verfahrens, verfolgt FLAM ein Konzept, das es erm glicht, Dateien so zu konvertieren, dass Kompatibilit tsforderungen weitgehend erf llbar sind. So ist die mit FLAM komprimierte Datei ein auf der Basis von Datens tzen logisches Abbild der urspr nglichen Datei. Davon ausgehend ist jede Konvertierung im Prinzip realisierbar.

Damit FLAM heterogen-kompatibel und hinsichtlich unterschiedlicher Anwendungsgebiete durchg ngig einsetzbar ist, wird das Komprimat, die FLAMFILE, in Anlehnung an das vorgenannte Prinzip standardm ig als sequentielle Datei abgelegt. F r Direktzugriffe ist auch eine Speicherung in einer indexsequentiellen Datei m glich.

Die Probleme, die bei vergleichbaren Anforderungen mit unkomprimierten Dateien auftreten, d rfen wegen des Einsatzes von FLAM deshalb nicht einfach ignoriert werden. Manche sind durch das FLAM-Konzept leichter zu l sen, andere bleiben trotz FLAM bestehen und m ssen daher, wie bisher, anwendungsspezifisch bzw. organisatorisch gel st werden, nur dass dabei die Originaldatei durch eine FLAMFILE ersetzt werden kann.

FLAM l st nicht die Probleme der heterogenen Kompatibilit t von Satz-/Feldstrukturen, die aus der Sicht eines Benutzers gegebenenfalls gar nicht erkannt werden. FLAM bietet hier zumindest Benutzerausg nge, um solche differenzierten Konvertierungen integrieren zu k nnen. Damit ist FLAM selbst offen f r L sungen, die sich in der Zukunft f r Teilbereiche standardisieren lassen.

FLAM verlangt, dass die zu komprimierenden Daten satzweise bergeben werden. Ferner bedingt das Verfahren ein asynchrones Vorgehen insofern, als aus n Originals tzen k Komprimatss tze mit n ungleich k werden k nnen. Das kann im Einzelfall ein Problem sein.

Die FLAMFILE wird grunds tzlich mit einer maximalen Satzl nge angelegt, die der Anwender selbst vorgeben kann. Das bewirkt in der Regel, das gleichlange Datens tze erzeugt werden. Dies ist erforderlich, weil es DV-Systeme gibt, die nur Dateien mit gleich langen S tzen unterst tzen. Diese Restriktion gilt zum Teil auch f r manche bertragungstechnik.

Die kleinste Satzl nge betr gt 80 Bytes, damit kann die FLAMFILE auch im Lochkarten-Format dargestellt werden (RJE-Filetransfer!). Die Begrenzungen nach oben richten sich danach, auf welchen Systemen die Datei gespeichert und mit welchen Produkten sie bertragen werden soll. Maximal sind 32760 Bytes m glich.

Unabhängig davon, kann der Anwender festlegen, welches Format der einzelne Satz haben soll: fix oder variabel. Dabei wird ein Komprimatssatz, der die maximale Satzlänge nicht ausfüllt, bei fixer Darstellung ggf. entsprechend aufgefüllt.

Ferner ist es möglich, Sätze unterschiedlich zu blocken, um das Ein-/Ausgabeverhalten sowie die Datenübertragung und/oder den Verbrauch an Speicherplatz zu optimieren.

Auch bezüglich Satzformat und Blockgröße können somit die Anforderungen aller beteiligten Hard- und Softwarekomponenten sowie spezifischer Anwendungen in der Regel auf einen Nenner gebracht werden.

Grundsätzlich ist die FLAMFILE eine binäre Datei, in der alle 256 Bitkombinationen je Byte erlaubt sind. In dieser Codierung kann die FLAMFILE nur transparent übertragen werden (MODE=CX8, VR8 oder ADC).

Falls auf 7-Bit-Leitungen übertragen werden muss, expandieren Filetransferprodukte solche Binärdateien so, dass garantiert ASCII-kompatible Formate entstehen. Manche Produkte machen aus jedem Halbbyte ein Byte, andere benutzen ein Verfahren, bei dem 3 Bytes nur auf 4 Bytes expandiert werden.

Sofern die zu komprimierenden Daten nur aus abdruckbaren Zeichen bestehen, erlaubt FLAM über den Parameter MODE=CX7 eine andere, ggf. zweckmäßigere Codierung des Komprimats. In diesem Fall werden alle Zeichen aus der Originaldatei direkt in das Komprimat übernommen. Es gibt keine Verschmelzung von Originalzeichen und FLAM-Deskriptoren. Diese Darstellung ist fast immer günstiger als die mit MODE=CX8 und anschließender Expansion im Verhältnis 3 zu 4.

Die FLAM-Deskriptoren selbst sind im MODE=CX7 ausschließlich solche abdruckbaren Zeichen, die international bezüglich ihrer Codierung in ASCII und EBCDIC eindeutig sind, und zwar alle großen und kleinen lateinischen Buchstaben, die zehn Ziffern und das Leerzeichen (Blank). Steuerzeichen, gleich welcher Art, Sonderzeichen, Umlaute usw. wurden ausgeschlossen.

Der Vorteil besteht nun darin, dass die im MODE=CX7 erstellte FLAMFILE an beliebiger Stelle zwischen Komprimierung und Dekomprimierung zeichenweise von ASCII nach EBCDIC oder umgekehrt 1:1 umcodiert werden kann. Wird diese Konvertierung nicht vom Übertragungssystem oder auf dem Übertragungsweg vorgenommen, kann der Anwender die Benutzerausgänge verwenden und die erforderliche Transkription als integrierten Pre-/Post-Prozess über die Sätze des Komprimats vornehmen.

FLAM orientiert sich bei MODE=CX7 immer an der normalen Arbeitsweise des Systems, auf dem FLAM installiert wurde. Sind Sende- und Zielsystem ASCII- oder EBCDIC-orientiert, er brigt sich jegliche Umcodierung. Sind Sende- und Zielsystem nicht gleich orientiert, erwartet FLAM zwingend, dass die FLAMFILE in der Codierung zur Dekomprimierung bergeben wird, die f r das Zielsystem charakteristisch ist.

Soll die mit MODE=CX7 erzeugte FLAMFILE sowohl ber 7-Bit- als auch 8-Bit-Leitung bertragen werden, sind differenzierte berlegungen anzustellen, um durchg ngig kompatibel zu bleiben. Dabei ist zu ber cksichtigen, dass FLAM die M glichkeit einer integrierten Codetransformation nicht auf allen Systemen anbietet. Im Grundsatz ist auch dieses Problem im CX7-Format l sbar.

Da die FLAMFILE in der Regel gleichlange S tze hat, wird der letzte Satz bei MODE=CX7 mit Blanks, sonst mit bin ren Nullen aufgef llt. Bei variablem Format wird er ggf. verk rzt.

Jeder Satz der FLAMFILE hat einen (internen) Overhead: die FLAM-Syntax. Damit wird das Komprimat in eine feste Struktur gebracht, die notwendig ist, um diversen Anforderungen zu gen gen. Der Overhead ist pro Satz gleich: Er betr gt im 7-Bit-Format 4 und im 8-Bit-Format 6 Bytes. Das sollte der Anwender wissen, wenn er die Satzl nge vordefiniert, insbesondere bei kurzen Komprimatss tzen. Dar ber hinaus gibt es weitere syntaktische Elemente in der FLAMFILE, z.B. je Original-Datei (optional) den Fileheader, je Matrix (obligatorisch) den Blockheader u.a.m.

Die FLAMFILE beginnt normalerweise mit einem Fileheader. Dieser besteht aus einem neutralen und einem systembezogenen Teil. Er beinhaltet in unterschiedlicher Ausf hrlichkeit die Informationen der zur Komprimierung zugewiesenen Original-Datei. Beim Dekomprimieren kann sich FLAM wahlweise dieser oder anderer, von au en vorgebbarer Informationen zum Aufbau der dekomprimierten Datei bedienen.

Es ist m glich, mehrere Komprimate zusammenzuf gen. Dann stehen in der FLAMFILE mehrere verschiedene Fileheader. Das Dienstprogramm FLAM bergeht diese beim Dekomprimieren und benutzt nur den Fileheader am Beginn der FLAMFILE. Die anderen werden aber protokolliert. Damit ist FLAM darauf vorbereitet, in archivierte Dateien identische Fileheader einzustreuen, um die Archivkopie auch bei Materialdefekten am Dateianfang noch identifizieren zu k nnen. ber die Satzschnittstelle k nnen die einzelnen Dateien getrennt werden.

Eine leere Datei wird in eine FLAMFILE konvertiert, die nur einen Header beinhaltet. Die Behandlung leerer Dateien ist damit kein Sonderfall mehr. Die blichen Probleme mit der Kommandosprache oder einem Filetransfer treten nicht mehr auf.

Beim Komprimieren kann ber Parameter bestimmt werden, ob und in welchem Umfang ein Fileheader erzeugt wird.

Um sich ber den Ursprung und die Eigenschaften eines Komprimats zu informieren, kann der Fileheader protokolliert werden, ohne dass die Datei dekomprimiert werden muss.

Je Matrix wird ein Blockheader gebildet. Dieser ist so aufgebaut, dass eine FLAMFILE auch ohne Fileheader korrekt dekomprimiert werden kann. Hier muss der Benutzer per Parameter, Kommandosprache oder Katalog mitteilen, in welches Format konvertiert werden soll, sofern ein anderes Format als sequentiell und variabel erzeugt werden soll.

Der Blockheader beinhaltet auch s mtliche Informationen, die FLAM zur Dekomprimierung braucht, z.B. MODE, Version, Matrixgr e u.a. Auf diese Weise wird die Aufw rtskompatibilit t von FLAM sichergestellt.

Die einzelnen S tze der FLAMFILE f hren ihre L nge redundant mit. Dazu kommt bei Darstellung im variablen Format das Satzl ngenfeld von 2 oder 4 Bytes L nge.

Auf PC- und UNIX-Systemen werden bei MODE=CX7 auch Texttrenner von 2 bzw. 1 Byte L nge benutzt. Insofern ist die Satzl nge heterogen als physikalische Gr e nicht eindeutig definiert.

Eine im 8-Bit-Code erstellte FLAMFILE wird pro Satz mit einer 16-Bit-Checksumme vor Datenverf lschung gesch tzt. Au erdem gibt es einen sogenannten Blockpointer, der eine Synchronisation erm glicht, falls Daten durch Verf lschung oder physischen Verlust nicht ordnungsgem   dekomprimiert werden k nnen.

Eine im 7-Bit-Code erstellte FLAMFILE beinhaltet keine Checksumme, da sie von ASCII nach EBCDIC und umgekehrt zeichenweise konvertierbar sein muss. Stattdessen wird gepr ft, ob es in der Anzahl Bytes je Satz eine Verschiebung gibt, z.B. weil die Code-Konvertierung nicht 1:1 erfolgte. Dies ist denkbar, wenn Tabulatoren oder Drucksteuerzeichen o. . nicht 1:1 umgesetzt werden. Dies widerspr che der Voraussetzung, dass nur solche Dateien mit MODE=CX7 bearbeitet werden d rfen, die aus abdruckbaren Zeichen bestehen.

Es ist von Vorteil im 8-Bit-Format zu arbeiten, wenn das 7-Bit-Format nicht zwingend erforderlich ist. Das geht schneller, der Kompressionsgrad ist h her, das Komprimat ist im Sinne von Datenschutz und Datensicherheit besser abgesichert, die bertragung solcher Dateien im Transparenzmodus ist effizienter und es gibt mehr Verschl sselungsm glichkeiten.

Eine FLAMFILE im 7-Bit-Code darf n mlich nur durch Verw rfelung von Zeichenfolgen zus tzlich verschleiert werden, wenn sie den sonstigen Anforderungen an dieses Format noch gen gen soll (siehe oben).

Eine FLAMFILE im 8-Bit-Format kann mit beliebigem Verfahren bearbeitet werden, um die FLAMFILE zur Marktversion hin gezielt inkompatibel zu machen.

F r den Fall, dass die unkomprimierten Datens tze vor der Komprimierung respektive nach der Dekomprimierung zeichenweise 1:1 umcodiert werden sollen, bietet FLAM die M glichkeit f r Konvertierungen von ASCII nach EBCDIC und umgekehrt, sowie von EBCDIC des einen Herstellers auf das eines anderen an. Diese Umsetztabellen von FLAM k nnen auch durch eigene Tabellen des Benutzers ersetzt werden. Es ist somit m glich, sie auf diese Weise auch zu Verschleierungszwecken zu benutzen. F r alle hier nicht aufgef hrten Konvertierungsprobleme kann der Anwender die Benutzerausg nge f r unkomprimierte Daten verwenden, und zwar unabh ngig vom MODE-Parameter. Diese k nnen zweckm igerweise mit Satzverarbeitungen kombiniert werden.

Unabh ngig von den Benutzerausg ngen gibt es die Satzschnittstelle zur bergabe unkomprimierter Datens tze vor dem Komprimieren bzw. nach dem Dekomprimieren. Diese erm glichen dem Anwender, Originaldateien zu verarbeiten, die FLAM nicht bearbeiten kann. Au erdem sind Kopplungen von FLAM mit Applikationen des Anwenders und anderen Produkten ber diese Satzschnittstelle m glich.

Auch wenn die FLAMFILE ohne Fileheader (HEADER=NO) geschrieben wurde, ist FLAM in der Lage, diese FLAMFILE zu dekomprimieren.

Die Restauration einer defekten FLAMFILE ist prinzipiell m glich und erfordert derzeitig die Hinzuziehung eines Spezialisten des Herstellers. Solche Defekte haben aber ihre Ursache ausschlie lich in Materialsch den sowie Datenverf lschungen des Komprimats von au en.

4.6.2 Sammeldatei

Die M glichkeit, mehrere Komprimate hintereinander abspeichern zu k nnen, wurde in der FLAMFILE als Sammeldatei weiterentwickelt.

Werden bei der Komprimierung mehrere Dateien gelesen (siehe Kapitel 3.1.4), so erzeugt FLAM f r jede Eingabedatei einen Fileheader (Parameter HEADER=YES, Standard) in der FLAMFILE. Praktisch werden so "viele FLAMFILEs" physikalisch sequentiell hintereinander geschrieben (Bei Parameter HEADER=NO werden keine Informationen ber die jeweilige Datei in der Sammeldatei gespeichert. Diese Datei wird dann bei der Dekomprimierung nicht mehr als FLAMFILE vieler Einzelkomprimate erkannt und kann dann auch nur insgesamt dekomprimiert werden.).

Dateityp und Format einer Sammeldatei k nnen, wie bei der FLAMFILE gewohnt, beliebig den W nschen angepasst werden.

ber die Parametereingabe SHOW=DIR lassen sich die Informationen aller komprimierten Dateien in dieser Sammeldatei anzeigen, ohne dass dekomprimiert wird.

FLAM kann bei der Dekomprimierung bei Vorgabe einer Auswahlvorschrift (siehe Kapitel 3.1.4.3) jede Datei dieser Sammeldatei dekomprimieren. Dabei kann die dekomprimierte Datei per Kommando vorgegeben werden, oder FLAM legt sie dynamisch an und katalogisiert sie.

Bibliotheken werden von FLAM memberweise in eine Sammeldatei komprimiert, d.h. jedes Member k nnte bei entsprechender Umsetzvorschrift in eine separate Datei dekomprimiert werden. Analog gilt die Umkehrung: aus vielen Einzeldateien k nnen Member einer Bibliothek erzeugt werden.

Durch diese Sammeldatei k nnen Bibliotheken verschiedenster Betriebssysteme heterogen kompatibel ausgetauscht werden.

Ohne Vorgabe einer Auswahl- oder Umsetzvorschrift wird wie in fr heren Versionen von FLAM in eine vorgegebene Datei dekomprimiert, d.h. alle urspr nglich verschiedenen Dateien stehen jetzt dekomprimiert hintereinander. Dabei wird gem den Dateiattributen der Ausgabe entsprechend konvertiert.

Hinweis: Wurde beim Erzeugen der Sammeldatei FILEINFO= NO angeben, so wurde auch kein Dateiname f r das jeweilige Komprimat gespeichert. Damit st nde auch kein Dateiname zum Anlegen der Dateien zur Verf gung.

ber die internen Dateinamen FILE0001 (f r die 1. Datei) bis FILE9999 (f r die 9999. Datei) k nnen die Komprimate

trotzdem angesprochen und entsprechende Umsetzvorschriften benannt werden.

4.7 Heterogener Datenaustausch

Komprimierte Dateien k nnen ber Filetransfer oder mit Hilfe von Datentr gern von einem System zu einem anderen gebracht werden. Dabei ist es nicht zwingend notwendig, dass es sich um gleichartige Systeme handelt. Voraussetzung ist nat rlich, dass ein Filetransfer f r den heterogenen Datenaustausch bzw. ein kompatibler Datentr ger vorhanden ist.

Unter den genannten Voraussetzungen ist ein Austausch von komprimierten Daten immer dann m glich, wenn auf den beteiligten Systemen FLAM existiert und installiert ist.

Zu beachten ist die Version von FLAM. F r BS2000, MVS und DOS/VSE gibt es eine Version 1.x von FLAM. Auf anderen Systemen gibt es FLAM erst ab Version 2.x. Die Versionen von FLAM sind aufw rtskompatibel. Das hei t, es k nnen auf Systemen mit Version 1.x und 2.x Komprimate, die auf einem beliebigen System mit Version 1.x erstellt wurden, dekomprimiert werden.

Ab Version 2.x sind die komprimierten Datenformate auf allen Systemen, auf denen es FLAM gibt, kompatibel.

F r den Datenaustausch zwischen gleichen und heterogenen Systemen sollten nur logische Datenformate f r die Komprimierung benutzt werden. Physische Formate sind auf einem anderen System nicht identisch reproduzierbar.

Es gibt mehrere Methoden f r die Erstellung eines Komprimates. Mit VR8 und CX8 werden Komprimate im 8-Bit Modus erstellt, mit CX7 im 7-Bit Modus. Nicht alle diese Methoden sind auf allen Rechnern implementiert. Bei einem Austausch von Dateien zwischen Gro rechnern kann jeder Modus benutzt werden.

Au erdem ist zu beachten, ob ein Filetransfer Daten transparent bertragen kann. In diesem Fall ist ein 8-Bit Komprimat, das auch im Zielsystem dekomprimiert werden kann, zu w hlen.

Bei nicht transparentem bertragungsmodus muss CX7 gew hlt werden. Die Datei darf nur druckbare Zeichen, die bei einer Code-Konvertierung im Filetransfer eindeutig umgesetzt werden, enthalten.

Beim Filetransfer sind au erdem bertragungsmodus, die Satzl nge und das Satzformat, variabel bzw. fix, zu beachten. Es ist m glich, dass im Zielsystem vor der Dekomprimierung L ngenfelder erg nzt oder gel scht werden m ssen. Einige Filetransfers erlauben z.B. nur bestimmte Satzl ngen oder Satzformate.

Ein Parameter, der auf beiden Systemen mit gleichem Wert vorhanden sein muss, ist die maximale Puffergr sse (MAXBUFFER) f r die Komprimierung eines Datenblockes. Dieser Wert betr gt bei Hostsystemen maximal 2,5 MB, bei einigen Rechnern, wie z.B. IBM/81xx und NIXDORF 886x konstant 32 KB, unter UNIX und UNIX-Derivaten max. 128 KB. FLAM auf Gro rechnern arbeitet mit Wechselpuffern so, dass f r die doppelte Puffergr e Speicherplatz zur Verf gung stehen muss.

Dateiattribute der Originaldateien sind beim Datenaustausch nicht von Bedeutung. bertragen wird das Komprimat als sequentielle Datei.

Im Zielsystem k nnen die dekomprimierten Daten in einer Datei, mit einer dort g ltigen Organisation, gespeichert werden. Diese kann einen sequentiellen, indexsequentiellen oder direkten Zugriff erlauben.

Wichtig ist, dass die Daten den Anforderungen der Organisation gen gen (z.B. muss ein Satzschl ssel f r index-sequentielle Organisation aufsteigend sortiert sein).

Dateien k nnen nach einer Verarbeitung komprimiert und bis zu einer bertragung komprimiert gespeichert oder erst unmittelbar vor einer bertragung komprimiert werden.

4.8 Code-Konvertierung

Bei der Komprimierung und Dekomprimierung k nnen beliebige 1:1 Code-Konvertierungen f r die Originaldaten durchgef hrt werden.

Eine Konvertierung von EBCDIC nach ASCII ist nach einer vorgegebenen Tabelle m glich. Es gibt aber auch die M glichkeit, eine eigene bersetzungstabelle mit der Angabe des Namens nachzuladen (TRANSLATE).

Generell ist es vorzuziehen, die Code-Konvertierung bei der Dekomprimierung durchzuf hren, weil das Komprimierungsverfahren bestimmte h ufige Zeichen (wie Leerzeichen und Nullen) des lokalen Zeichensatzes bevorzugt behandelt. Durch eine Transformation k nnte die Komprimierung verschlechtert werden. Au erdem ist bei einer Umsetzung von EBCDIC nach ASCII, wegen des kleineren Zeichenvorrates der Verlust von Zeichen m glich, die dann bei der Dekomprimierung nicht mehr in EBCDIC zur ck konvertiert werden k nnen.

Ein besonderes Problem ist der Zeichencode beim Austausch von Komprimaten indexsequentieller Dateien. Durch die Konvertierung alphanumerischer oder bin rer Schl ssel sind diese nach der Konvertierung nicht mehr sortiert. Keine Probleme gibt es bei abdruckbar alphabetischen oder abdruckbar numerischen Schl sseln.

Bei bin ren bzw. alphanumerischen Schl sseln ist eine Konversion der indexsequentiellen Datei vor bzw. nach der Verarbeitung mit FLAM notwendig.

4.9 Umsetzung von Dateiformaten

Dateien m ssen beim Dekomprimieren nicht mit der gleichen Organisation und dem gleichen Satzformat wie die Originaldatei erstellt werden. Das gilt insbesondere f r Komprimate von anderen Betriebssystemen.

Wenn keine anderen Angaben vom Anwender gemacht werden, werden Dateien, die unter dem gleichen Betriebssystem komprimiert wurden, durch die Angaben im systemspezifischen Teil des Fileheaders mit den gleichen Attributen rekonstruiert.

Grunds tzlich ist jedoch jedes Komprimat in jedes Dateiformat konvertierbar, das von FLAM auf dem jeweiligen System unterst tzt wird.

Dabei k nnen in Abh ngigkeit von der Dateiorganisation und dem Satzformat verschiedene Situationen auftreten:

Bei der Umsetzung in fixes Satzformat k nnen die Originaldaten l nger oder k rzer als die neue Satzl nge sein.

L ngere Originaldaten k nnen durch den Parameter TRUNCATE=YES auf Anforderung verk rzt werden.

K rzere Originaldaten werden bis zur neuen (fixen) Satzl nge mit F llzeichen (PADCHAR) aufgef llt.

Beim Umsetzen von indexsequentiellen Dateien in sequentielle Dateien, k nnen durch den Parameter KEYDISP=DEL die Schl ssel entfernt werden.

Beim Umsetzen von sequentiellen Dateien in ein indexsequentielles Format m ssen die Originaldaten ein Feld mit einer Schl sseleigenschaft (eindeutig und aufsteigend sortiert) enthalten. Anderenfalls kann mit dem Parameter KEYDISP=NEW ein abdruckbarer Schl ssel in der gew nschten L nge an der Schl sselposition eingef gt werden.

S tze der L nge Null oder L cken aus relativen Dateien werden beim Konvertieren in ein sequentielles Format entfernt.

Beim Umsetzen in fixes Format werden L cken entfernt.

FLAM (VSE/ESA)

Benutzerhandbuch

Kapitel 5:

Anwendungsbeispiele

Inhalt

5. Anwendungsbeispiele

Nachfolgend sind einige Beispiele zur Demonstration unterschiedlicher FLAM-Funktionen angegeben. Alle Beispiele sind in Form von Kommandoprozeduren oder Quelltexten auf dem Lieferband enthalten.

Die Beispiele sind alle getestet. Trotzdem ist es m glich, dass einzelne Beispiele in anderen Umgebungen nicht in jedem Falle ohne Probleme ablauff hig und Anpassungen notwendig sind.

Bei den COBOL-Programmen wurde versucht, m glichst unabh ngig von Compiler und Betriebssystem zu bleiben. Die Programme wurden deshalb sowohl auf MVS als auch auf BS2000 und DPPX getestet. Beim Portieren von MVS auf BS2000 und DPPX mussten dabei einige Modifikationen gemacht werden.

5.1 JCL f r Dienstprogramm FLAM

Es folgen Jobabl ufe f r die Komprimierung und Dekomprimierung

5.1.1 Komprimieren

Beispiel: **Komprimieren mit Zuweisung der Ein-/Ausgabedatei ber JCL.**

```
* $$ JOB JNM=JFLAMKV,CLASS=0,DISP=D,PRI=3,NTFY=YES,LDEST=*          (0)
* $$ LST CLASS=A,DISP=D                                             (1)
// JOB JFLAMKV TESTJOB FUER FLAM                                    (2)
// STDOPT LINES=66
// OPTION PARTDUMP
// LIBDEF PHASE,SEARCH=(FLAM.LIB)                                   (3)
* TESTJOB KOMPRIMIEREN MIT FLAM INPUT(VSAM) OUTPUT(VSAM)
* *****************************************************************
* ** ZUWEISUNG VON VSAM UEBER JCL.                                **
* *****************************************************************
// DLBL FLAMIN,'LIMES.FLAM.DATEN',,VSAM,CAT=VSESPUC                 (4)
// DLBL FLAMFIL,'LIMES.FLAMFILE',,VSAM,CAT=VSESPUC                  (5)
// EXEC FLAM,SIZE=AUTO,PARM='COMP,MODE=CX8,MAXSIZE=2048,END'        (6)
/*                                                                  (7)
/&                                                                  (8)
* $$ EOJ                                                            (9)
```

(0) POWER Job-Karte.

(1) POWER Listen Zuweisung f r den Drucker.

(2) JCL-JOB-Anweisung.

(3) Zuweisung der Ladebibliothek

(4) Beschreibung der Eingabedatei mit den Originaldaten. Hier eine VSAM-ESDS-Datei. Die Dateiattribute werdendem Katalog entnommen.

(5) Beschreibung der Komprimatsdatei FLAMFILE als VSAM-ESDS-Datei. Als Re cordgr e wird die MAXSIZE-Angabe aus der PARM Anweisung verwendet, falls diesen nicht gr er ist als die maximale Recordgr e laut Katalog.

(6) EXEC-Anweisung f r FLAM mit PARM-Angabe zum komprimieren.

(7) EOF fuer READER-Input (immer erforderlich).

(8) JOB-Ende

(9) POWER-JOB-Ende

Protokoll:

```
// JOB JFLAMKV TESTJOB FUER FLAM
// STDOPT LINES=66
// OPTION PARTDUMP
// LIBDEF PHASE,SEARCH=(FLAM.LIB)
* TESTJOB KOMPRIMIEREN MIT FLAM INPUT(VSAM) OUTPUT(VSAM)
* *******************************************************************
* ** ZUWEISUNG VON VSAM UEBER JCL.                                 **
* *******************************************************************
// DLBL FLAMIN,'LIMES.FLAM.DATEN',,VSAM,CAT=VSESPUC
// DLBL FLAMFIL,'LIMES.FLAMFILE',,VSAM,CAT=VSESPUC
// EXEC FLAM,SIZE=AUTO,PARM='COMP,MODE=CX8,MAXSIZE=2048,END'

 FLM0448 COPYRIGHT (C) 1989-1999 BY LIMES DATENTECHNIK TS2000-06-30
 FLM0428 RECEIVED: COMP,MODE=CX8,MAXSIZE=2048,END
 FLM0400 FLAM COMPRESSION VERSION 3.0A00 ACTIVE
 FLM0410 DATA SET NAME : VSESPUC:LIMES.FLAM.DATEN  -FLAMIN-
 FLM0415 USED PARAMETER: ACCESS  : LOG
 FLM0415 USED PARAMETER: IDSORG  : ESDS
 FLM0415 USED PARAMETER: IRECFORM: SPNBLK
 FLM0415 USED PARAMETER: IRECSIZE:    32758
 FLM0415 USED PARAMETER: IBLKSIZE:     2048
 FLM0410 DATA SET NAME : VSESPUC:LIMES.FLAMFILE  -FLAMFILE-
 FLM0415 USED PARAMETER: MODE    : CX8
 FLM0415 USED PARAMETER: MAXBUFF :    65536
 FLM0415 USED PARAMETER: MAXREC  :      255
 FLM0415 USED PARAMETER: MAXSIZE :     2048
 FLM0415 USED PARAMETER: DSORG   : ESDS
 FLM0415 USED PARAMETER: RECFORM : VARBLK
 FLM0415 USED PARAMETER: BLKSIZE :     2560
 FLM0406 INPUT   RECORDS/BYTES:       8,511 /              680,880
 FLM0407 OUTPUT  RECORDS/BYTES:         235 /              481,280
 FLM0416 COMPRESSION REDUCTION IN PERCENT:    29.32
 FLM0408 CPU - TIME:        2.6632
 FLM0409 RUN - TIME:        7.4824
 FLM0440 FLAM COMPRESSION NORMAL END
1S55I  LAST RETURN CODE WAS 0000
EOJ JFLAMKV   MAX.RETURN CODE=0000
```

Beispiel: **Komprimieren mit Zuweisung der Eingabedatei ber Parameter**

```
* $$ JOB JNM=JTSVSAM1,CLASS=0,DISP=D,PRI=3,NTFY=YES,LDEST=*          (0)
* $$ LST DISP=D,CLASS=A,PRI=3                                         (1)
// JOB JFV30KV TESTJOB FUER FLAM                                      (2)
// STDOPT LINES=66
// OPTION PARTDUMP
// LIBDEF PHASE,SEARCH=(FLAM.LIB)                                     (3)
* TESTJOB KOMPRIMIEREN MIT FLAM INPUT(VSAM) OUTPUT(VSAM)
* *****************************************************************
* ** ZUGRIFF AUF VSAM DATA-SET UEBER DSN.                        **
* *****************************************************************
// DLBL FLAMFIL,'LIMES.FLAMFILE',,VSAM,CAT=VSESPUC                    (4)
// EXEC FLAM,SIZE=AUTO                                                (5)
COMP,FLAMIN=VSESPUC:LIMES.FLAM.DATEN,MODE=ADC,END                     (6)
/*                                                                    (7)
/&                                                                    (8)
* $$ EOJ                                                              (9)
```

(0) POWER Job-Karte.

(1) POWER Listen Zuweisung f r den Drucker.

(2) JCL-JOB-Anweisung.

(3) Zuweisung der Ladebibliothek

(4) Beschreibung der Komprimatsdatei FLAMFILE als VSAM-ESDS-Datei. Als Re cordgr e wird die MAXSIZE-Angabe aus den Defaultwerten verwendet, falls diesen nicht gr er ist als die maximale Recordgr e laut Katalog. Die Dateiattribute werden dem Katalog entnommen.

(5) EXEC-Anweisung f r FLAM.

(6) Parameter zum Komprimieren mit Komprimierungsmode ADC. Die Eingabedatei mit den Originaldaten wird ber den Dateinamen zugeordnet. Die Dateiat- tribute werdendem Katalog entnommen.

(7) EOF fuer READER-Input (immer erforderlich).

(8) JOB-Ende

(9) POWER-JOB-Ende

Protokoll:

```
// JOB JFV30KV TESTJOB FUER FLAM
// STDOPT LINES=66
// OPTION PARTDUMP
// LIBDEF PHASE,SEARCH=(FLAM.LIB)
* TESTJOB KOMPRIMIEREN MIT FLAM INPUT(VSAM) OUTPUT(VSAM)
* *****************************************************************
* ** ZUGRIFF AUF VSAM DATA-SET UEBER DSN.                         **
* *****************************************************************
// DLBL FLAMFIL,'LIMES.FLAMFILE',,VSAM,CAT=VSESPUC
// EXEC FLAM,SIZE=AUTO

 FLM0448 COPYRIGHT (C) 1989-1999 BY LIMES DATENTECHNIK TS2000-06-30
 FLM0428 RECEIVED: COMP,FLAMIN=VSESPUC:LIMES.FLAM.DATEN,MODE=ADC,END
 FLM0400 FLAM COMPRESSION VERSION 3.0A00 ACTIVE
 FLM0410 DATA SET NAME : VSESPUC:LIMES.FLAM.DATEN  -FLAMIN-
 FLM0415 USED PARAMETER: ACCESS  : LOG
 FLM0415 USED PARAMETER: IDSORG  : ESDS
 FLM0415 USED PARAMETER: IRECFORM: SPNBLK
 FLM0415 USED PARAMETER: IRECSIZE:    32758
 FLM0415 USED PARAMETER: IBLKSIZE:     2048
 FLM0410 DATA SET NAME : VSESPUC:LIMES.FLAMFILE  -FLAMFILE-
 FLM0415 USED PARAMETER: MODE    : ADC
 FLM0415 USED PARAMETER: MAXBUFF :    65536
 FLM0415 USED PARAMETER: MAXREC  :     4095
 FLM0415 USED PARAMETER: MAXSIZE :      512
 FLM0415 USED PARAMETER: DSORG   : ESDS
 FLM0415 USED PARAMETER: RECFORM : VARBLK
 FLM0415 USED PARAMETER: BLKSIZE :     2560
 FLM0406 INPUT   RECORDS/BYTES:       8,511 /              680,880
 FLM0407 OUTPUT  RECORDS/BYTES:         473 /              242,176
 FLM0416 COMPRESSION REDUCTION IN PERCENT:    64.44
 FLM0408 CPU - TIME:         9.3363
 FLM0409 RUN - TIME:        13.3658
 FLM0440 FLAM COMPRESSION NORMAL END
1S55I  LAST RETURN CODE WAS 0000
EOJ JFV30KV   MAX.RETURN CODE=0000
```

Beispiel: **Komprimieren einer bestimmten Liste in der POWER List-Queue, es wird keine FLAMFILE erzeugt (z.B. zum Testen).**

```
// JOB JFCPWR1 TESTJOB ZUR KOMPRIMIERUNG MIT FLAM
// STDOPT LINES=66
// OPTION PARTDUMP
// LIBDEF PHASE,SEARCH=(FLAM.LIB)
* ** KOMPRIMIERN MIT FLAM: POWER QUEUE * *
* SYNTAX: FLAMIN=#LST/#PUN/#RDR(JOBNAME,JOBNR,CLASS,DISP,USER,PASSW)
// EXEC FLAM,SIZE=AUTO

1S54I  PHASE FLAM     IS TO BE FETCHED FROM FLAM.LIB
 FLM0448 COPYRIGHT (C) 1989-2007 BY LIMES DATENTECHNIK TS2009-06-30
 FLM0428 RECEIVED: COMPRESS,MODE=ADC
 FLM0428 RECEIVED: FLAMIN=#LST(JFCMP,3569,A,D,FLAM)               (1)
 FLM0428 RECEIVED: FLAMFILE=*DUMMY                                (2)
 FLM0400 FLAM COMPRESSION VERSION 4.1A00 ACTIVE
 FLM0410 DATA SET NAME : #LST(JFCMP,3569,A,D,FLAM,)  -FLAMIN-
 FLM0415 USED PARAMETER: IDSORG  : SEQUENT
 FLM0415 USED PARAMETER: IRECFORM: VAR
 FLM0415 USED PARAMETER: IRECSIZE:    32756
 FLM0415 USED PARAMETER: IBLKSIZE:    32764
 FLM0415 USED PARAMETER: IPRCNTRL: MACHINE
 FLM0410 DATA SET NAME : (NONE)    -FLAMFILE-                     (3)
 FLM0415 USED PARAMETER: MODE    : ADC
 FLM0415 USED PARAMETER: MAXBUFF :    65536
 FLM0415 USED PARAMETER: MAXREC  :     4095
 FLM0415 USED PARAMETER: MAXSIZE :      512
 FLM0415 USED PARAMETER: DSORG   : SEQUENT
 FLM0415 USED PARAMETER: RECFORM : VARBLK
 FLM0415 USED PARAMETER: BLKSIZE :     6144
 FLM0406 INPUT   RECORDS/BYTES:            33 /              1,618
 FLM0407 OUTPUT  RECORDS/BYTES:             2 /              1,024
 FLM0416 COMPRESSION REDUCTION IN PERCENT:    36.72
 FLM0408 CPU - TIME:        0.0240
 FLM0409 RUN - TIME:        2.9986
 FLM0440 FLAM COMPRESSION NORMAL END
1S55I  LAST RETURN CODE WAS 0000
```

(1) Spezielle Liste (Dateiname gem FLAM-Syntax, siehe Kap. 3.1.4)
(2) Die FLAMFILE wird auf *DUMMY gesetzt, damit erfolgt keine Ausgabe
(3) Kein Dateiname (wg. 2), Ausgabe nur virtuell aber kompl. Protokoll

Beispiel: **Komprimieren aller Listen der POWER List-Queue, serieller Splitt der FLAMFILE in Dateien von 1MB Gr e.**

Mit Meldung FLM0414 wird der Splitt der FLAMFILE best tigt. Meldung FLM0468 protokolliert die Anzahl S tze/Byte pro erzeugtem FLAMFILE-Fragment. Der zugeh rige Dateiname wird in der vorherigen Meldung FLM0410 ausgegeben. Am Protokollende wird eine Gesamtstatistik ausgegeben.

```
// JOB JFCSPLI TESTJOB ZUR KOMPRIMIERUNG MIT FLAM
// STDOPT LINES=66
// OPTION PARTDUMP
// LIBDEF PHASE,SEARCH=(FLAM.LIB)
* SPLITT DER FLAMFILE
// DLBL FLAMFIL,'DAT.WORK01.ESDS',,VSAM,CAT=LIMES
// EXEC FLAM,SIZE=AUTO
1S54I  PHASE FLAM     IS TO BE FETCHED FROM FLAM.LIB
 FLM0448 COPYRIGHT (C) 1989-2007 BY LIMES DATENTECHNIK TS2009-06-30
 FLM0428 RECEIVED: COMPRESS,MODE=ADC
 FLM0428 RECEIVED: FLAMIN=#LST(*)
 FLM0428 RECEIVED: SPLITM=SER,SPLITS=1
 FLM0400 FLAM COMPRESSION VERSION 4.1A00 ACTIVE
 FLM0410 DATA SET NAME : #LST(FLAMDYN,2331,0,D,SYSA,)  -FLAMIN-
 FLM0415 USED PARAMETER: IDSORG  : SEQUENT
 FLM0415 USED PARAMETER: IRECFORM: VAR
 FLM0415 USED PARAMETER: IRECSIZE:    32756
 FLM0415 USED PARAMETER: IBLKSIZE:    32764
 FLM0415 USED PARAMETER: IPRCNTRL: MACHINE
 FLM0414 FLAMFILE SPLIT ACTIVE
 FLM0410 DATA SET NAME : LIMES:DAT.WORK01.ESDS  -FLAMFILE-
 FLM0415 USED PARAMETER: SPLITMOD: SERIAL
 FLM0415 USED PARAMETER: SPLITSIZ:        1
 FLM0415 USED PARAMETER: MODE    : ADC
 FLM0415 USED PARAMETER: MAXBUFF :    65536
 FLM0415 USED PARAMETER: MAXREC  :     4095
 FLM0415 USED PARAMETER: MAXSIZE :      512
 FLM0415 USED PARAMETER: DSORG   : ESDS
 FLM0415 USED PARAMETER: RECFORM : VARBLK
 FLM0415 USED PARAMETER: BLKSIZE :    20480
 FLM0406 INPUT   RECORDS/BYTES:            627 /              46,651
 FLM0407 OUTPUT  RECORDS/BYTES:             22 /              11,264
 FLM0410 DATA SET NAME : #LST(FLAMPAR,3037,0,D,FLAM,)  -FLAMIN-
 FLM0415 USED PARAMETER: IDSORG  : SEQUENT
 FLM0415 USED PARAMETER: IRECFORM: VAR
 FLM0415 USED PARAMETER: IRECSIZE:    32756
 FLM0415 USED PARAMETER: IBLKSIZE:    32764
 FLM0415 USED PARAMETER: IPRCNTRL: MACHINE
.
.
.
 FLM0468 SPLIT RECORDS/BYTES:          2,048 /           1,048,576
 FLM0410 DATA SET NAME : LIMES:DAT.WORK02.ESDS  -F882761 -
 FLM0406 INPUT   RECORDS/BYTES:          7,641 /             751,324
 FLM0407 OUTPUT  RECORDS/BYTES:            273 /             139,776
 FLM0410 DATA SET NAME : #LST(FLAMFIOP,3564,0,D,FLAM,)  -FLAMIN-
 FLM0415 USED PARAMETER: IDSORG  : SEQUENT
.
.
```

```
.
 FLM0410 DATA SET NAME : #LST(FLAMFIOV,3693,0,D,FLAM,)  -FLAMIN-
 FLM0415 USED PARAMETER: IDSORG  : SEQUENT
 FLM0415 USED PARAMETER: IRECFORM: VAR
 FLM0415 USED PARAMETER: IRECSIZE:    32756
 FLM0415 USED PARAMETER: IBLKSIZE:    32764
 FLM0415 USED PARAMETER: IPRCNTRL: MACHINE
 FLM0468 SPLIT RECORDS/BYTES:         2,048 /           1,048,576
 FLM0410 DATA SET NAME : LIMES:DAT.WORK03.ESDS  -F882761 -
 FLM0406 INPUT   RECORDS/BYTES:          9,833 /           1,011,708
 FLM0407 OUTPUT  RECORDS/BYTES:            321 /             164,352
.
.
.
 FLM0415 USED PARAMETER: IPRCNTRL: MACHINE
 FLM0406 INPUT   RECORDS/BYTES:             12 /                 742
 FLM0407 OUTPUT  RECORDS/BYTES:              1 /                 512
 FLM0468 SPLIT RECORDS/BYTES:         1,612 /             825,344
 FLM0410 DATA SET NAME : LIMES:DAT.WORK01.ESDS  -FLAMFILE-
 FLM0406 INPUT   RECORDS/BYTES:        242,151 /          18,369,031
 FLM0407 OUTPUT  RECORDS/BYTES:          5,702 /           2,919,424
 FLM0416 COMPRESSION REDUCTION IN PERCENT:    84.11
 FLM0408 CPU - TIME:        6.1277
 FLM0409 RUN - TIME:       12.2130
 FLM0440 FLAM COMPRESSION NORMAL END
1S55I  LAST RETURN CODE WAS 0000
```

5.1.2 Dekomprimieren

Beispiel: **Komprimieren mit Zuweisung der Ein-/Ausgabedatei ber JCL.**

```
* $$ JOB JNM=JFLAMDV,CLASS=0,DISP=D,PRI=3,NTFY=YES,LDEST=*          (0)
* $$ LST CLASS=A,DISP=D                                              (1)
// JOB JFLAMDV TESTJOB FUER FLAM                                     (2)
// STDOPT LINES=66
// OPTION PARTDUMP
// LIBDEF PHASE,SEARCH=(FLAM.LIB)                                    (3)
* TESTJOB DEKOMPRIMIEREN MIT FLAM INPUT(VSAM) OUTPUT(VSAM)
* ******************************************************************
* ** ZUWEISUNG VON VSAM UEBER JCL.                                **
* ******************************************************************
// DLBL FLAMOUT,'LIMES.FLAM.DATEN',,VSAM,CAT=VSESPUC                 (4)
// DLBL FLAMFIL,'LIMES.FLAMFILE',,VSAM,CAT=VSESPUC                   (5)
// EXEC FLAM,SIZE=AUTO,PARM='DECOMP,END'                             (6)
/*                                                                   (7)
/&                                                                   (8)
* $$ EOJ                                                             (9)
```

(0) POWER Job-Karte.

(1) POWER Listen Zuweisung f r den Drucker.

(2) JCL-JOB-Anweisung.

(3) Zuweisung der Ladebibliothek

(4) Beschreibung der Ausgabedatei f r die Originaldaten. Hier eine VSAM-ESDS-Datei. Die Dateiattribute werdendem Katalog entnommen.

(5) Beschreibung der Komprimatsdatei FLAMFILE. Die Dateiangaben werde dem Katalog entnommen.

(6) EXEC-Anweisung f r FLAM mit PARM-Angabe zum dekomprimieren.

(7) EOF fuer READER-Input (immer erforderlich).

(8) JOB-Ende

(9) POWER-JOB-Ende

Protokoll:

```
// JOB JFLAMDV TESTJOB FUER FLAM
// STDOPT LINES=66
// OPTION PARTDUMP
// LIBDEF PHASE,SEARCH=(FLAM.LIB)
* TESTJOB DEKOMPRIMIEREN MIT FLAM INPUT(VSAM) OUTPUT(VSAM)
* ******************************************************************
* ** ZUWEISUNG VON VSAM UEBER JCL.                                 **
* ******************************************************************
// DLBL FLAMOUT,'LIMES.FLAM.DATEN',,VSAM,CAT=VSESPUC
// DLBL FLAMFIL,'LIMES.FLAMFILE',,VSAM,CAT=VSESPUC
// EXEC FLAM,SIZE=AUTO,PARM='DECOMP,END'

 FLM0448 COPYRIGHT (C) 1989-1999 BY LIMES DATENTECHNIK TS2000-06-30
 FLM0428 RECEIVED: DECOMP,END
 FLM0450 FLAM DECOMPRESSION VERSION 3.0A00 ACTIVE
 FLM0460 DATA SET NAME : VSESPUC:LIMES.FLAMFILE  -FLAMFILE-
 FLM0465 USED PARAMETER: MODE    : ADC
 FLM0465 USED PARAMETER: VERSION :      300
 FLM0465 USED PARAMETER: FLAMCODE: EBCDIC
 FLM0465 USED PARAMETER: MAXBUFF :    65536
 FLM0465 USED PARAMETER: DSORG   : ESDS
 FLM0465 USED PARAMETER: RECFORM : VARBLK
 FLM0465 USED PARAMETER: RECSIZE :      512
 FLM0465 USED PARAMETER: BLKSIZE :     2560
 FLM0482 OLD ODSN    : VSESPUC:LIMES.FLAM.DATEN
 FLM0482 OLD ODSORG  : SEQUENT
 FLM0482 OLD ORECFORM: SPNBLK
 FLM0482 OLD ORECSIZE:    32758
 FLM0482 OLD OBLKSIZE:     2048
 FLM0469 COMPRESSED FILE FLAM-ID: 0102
 FLM0460 DATA SET NAME : VSESPUC:LIMES.FLAM.DATEN  -FLAMOUT-
 FLM0479 FILE-ATTRIBUTE WAS CHANGED
 FLM0480 FILE PARAM   OLD  ODSORG  : SEQUENT    NEW: ESDS
 FLM0456 INPUT   RECORDS/BYTES:         473 /             242,176
 FLM0457 OUTPUT  RECORDS/BYTES:       8,511 /             680,880
 FLM0458 CPU - TIME:        5.1593
 FLM0459 RUN - TIME:        9.6482
 FLM0490 FLAM DECOMPRESSION NORMAL END
1S55I  LAST RETURN CODE WAS 0000
EOJ JFLAMDV   MAX.RETURN CODE=0000
```

Beispiel: **Dekomprimieren mit DTFSD Ein- und Ausgabedatei.**

```
* $$ JOB JNM=JFLAMDSD,CLASS=0,DISP=D,PRI=3,NTFY=YES,LDEST=*            (0)
* $$ LST CLASS=A,DISP=D                                                (1)
// JOB JFLAMDSD TESTJOB FUER FLAM DTFSD > DTFSD                        (2)
// OPTION PARTDUMP
// STDOPT LINES=66
// LIBDEF PHASE,SEARCH=FLAM.LIB                                        (3)
* ** TESTJOB FUER FLAM DTFSD > DTFSD
* *******************************************************************
* ** ZUWEISUNG VON DTFSD ODER DTFMT NUR UEBER JCL MOEGLICH.        **
* *******************************************************************
// ASSGN SYS001,142,TEMP                                               (4)
// DLBL FLAMFIL,'FLAMFILE.TEST2',,SD,BLKSIZE=2560                      (5)
// ASSGN SYS002,142,TEMP                                               (6)
// DLBL FLAMOUT,'FLAMV30.TESTDATA',0,SD,BLKSIZE=1032                   (7)
// EXTENT SYS002,SYSWK2,1,,400064,20000                                (8)
// EXEC FLAM,SIZE=AUTO,PARM='DECOMP,RECFORM=FIX,MAXSIZE=512,END'       (9)
/*                                                                    (10)
/&                                                                    (11)
* $$ EOJ                                                              (12)
```

(0) POWER Job-Karte.

(1) POWER Listen Zuweisung f r den Drucker.

(2) JCL-JOB-Anweisung.

(3) Zuweisung der Ladebibliothek

(4) DEVICE-Zuordnung f r die FLAMFILE.

(5) Beschreibung der Komprimatsdatei FLAMFILE.

(6) DEVICE-Zuordnung f r die Ausgabedatei.

(7) Beschreibung der Ausgabedatei f r die Originaldaten im DTFSD-Format.

(8) Bereichszuweisung f r Ausgabedatei.

(9) EXEC-Anweisung f r FLAM mit PARM-Angabe zum dekomprimieren.

(10) EOF fuer READER-Input (immer erforderlich).

(11) JOB-Ende

(12) POWER-JOB-Ende

Protokoll:

```
// JOB JFLAMDSD TESTJOB FUER FLAM DTFSD > DTFSD
// OPTION PARTDUMP
// STDOPT LINES=66
// LIBDEF PHASE,SEARCH=FLAMV30.LIB
* ** TESTJOB FUER FLAM DTFSD > DTFSD
* *******************************************************************
* ** ZUWEISUNG VON DTFSD ODER DTFMT NUR UEBER JCL MOEGLICH.         **
* *******************************************************************
// ASSGN SYS001,142,TEMP
// DLBL FLAMFIL,'FLAMFILE.TEST2',,SD,BLKSIZE=2560
// ASSGN SYS002,142,TEMP
// DLBL FLAMOUT,'FLAMV30.TESTDATA',0,SD,BLKSIZE=1032
// EXTENT SYS002,SYSWK2,1,,400064,20000
// EXEC FLAM,SIZE=AUTO,PARM='DECOMP,RECFORM=FIX,MAXSIZE=512,END'

 FLM0448 COPYRIGHT (C) 1989-1999 BY LIMES DATENTECHNIK TS2000-06-30
 FLM0428 RECEIVED: DECOMP,RECFORM=FIX,MAXSIZE=512,END
 FLM0450 FLAM DECOMPRESSION VERSION 3.0A00 ACTIVE
 FLM0460 DATA SET NAME : FLAMFILE.TEST2  -FLAMFILE-
 FLM0465 USED PARAMETER: MODE    : ADC
 FLM0465 USED PARAMETER: VERSION :      300
 FLM0465 USED PARAMETER: FLAMCODE: EBCDIC
 FLM0465 USED PARAMETER: MAXBUFF :    65536
 FLM0465 USED PARAMETER: DSORG   : SEQUENT
 FLM0465 USED PARAMETER: RECFORM : FIX
 FLM0465 USED PARAMETER: RECSIZE :      512
 FLM0465 USED PARAMETER: BLKSIZE :     2560
 FLM0482 OLD ODSN    : VSESPUC:LIMES.FLAM.DATEN
 FLM0482 OLD ODSORG  : SEQUENT
 FLM0482 OLD ORECFORM: SPNBLK
 FLM0482 OLD ORECSIZE:    32758
 FLM0482 OLD OBLKSIZE:     2048
 FLM0469 COMPRESSED FILE FLAM-ID: 0102
 FLM0460 DATA SET NAME : FLAMV30.TESTDATA  -FLAMOUT-
 FLM0479 FILE-ATTRIBUTE WAS CHANGED
 FLM0480 FILE PARAM   OLD  OBLKSIZE:     2048   NEW:     1032
 FLM0456 INPUT   RECORDS/BYTES:         473 /             242,176
 FLM0457 OUTPUT  RECORDS/BYTES:       8,511 /             680,880
 FLM0458 CPU - TIME:        4.2396
 FLM0459 RUN - TIME:        8.1953
 FLM0490 FLAM DECOMPRESSION NORMAL END
1S55I  LAST RETURN CODE WAS 0000
EOJ JFLAMDSD  MAX.RETURN CODE=0000
```

5.2 Verwendung der Satzschnittstelle

Es folgen Beispielprogramme zum Aufruf der FLAM-Satzschnittstelle.

5.2.1 Komprimieren

Die sequentielle Datei "INDAT" mit fixer Satzl nge wird mit COBOL gelesen. Jeder Datensatz wird an die Satzschnittstelle bergeben. FLAM erzeugt die komprimierte FLAMFILE, die im n chsten Beispiel wieder gelesen wird.

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.  SAMPLE1C.
 AUTHOR.      LIMES DATENTECHNIK GMBH.
*
* SAMPLE1C READS A SEQUENTIAL DATA SET.
*         EVERY RECORD IS GIVEN TO FLAM FOR COMPRESSION.
*         FLAM MANAGES THE FLAMFILE ITSELF.
*
*         IN THIS EXAMPLE, THE FLAMFILE CAN BE
*            - ANY DATA SET   IN  MVS, BS2000
*            - VSAM               VSE/ESA
*
*         EINE SEQUENTIELLE DATEI WIRD GELESEN.
*         JEDER DATENSATZ WIRD AN FLAM ZUR KOMPRIMIERUNG
*         UEBERGEBEN.
*         FLAM VERWALTET DIE KOMPRIMATSDATEI SELBST.
*
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
*
 SPECIAL-NAMES.           SYSOUT IS  OUT-PUT.
*
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
     SELECT  INDAT  ASSIGN TO SYS010-S-DATAIN
             ACCESS MODE IS SEQUENTIAL
             ORGANIZATION IS SEQUENTIAL.
*
 DATA DIVISION.
*
 FILE SECTION.
 FD  INDAT   RECORD CONTAINS 80 CHARACTERS
             RECORDING MODE IS F.
*
 01  INDAT-RECORD.
     02  FILLER    PIC X(80).
*
 WORKING-STORAGE SECTION.
*
 77  OPERATION       PIC X(6).
*

 01  FLAM-PARAMETER.
```

```
      *
      *  USED FOR EVERY FLAM-CALL
      *
          02  FILE-ID   PIC S9(8)  COMP SYNC.
          02  RETCO     PIC S9(8)  COMP SYNC.
             88 FLAMOK  VALUE  0.
      *
          02  RETCO-X   REDEFINES RETCO.
            03 RETCO-1   PIC X.
             88 NODMS-ERROR   VALUE  LOW-VALUE.
            03 RETCO-2-4 PIC XXX.
      *
      *  USED FOR FLAM OPEN
      *
          02  LASTPAR   PIC S9(8)  COMP SYNC VALUE 0.
          02  OPENMODE  PIC S9(8)  COMP SYNC VALUE 1.
          02  DDNAME    PIC X(8)   VALUE "FLAMFIL".
          02  STATIS    PIC S9(8)  COMP SYNC VALUE 0.
      *
      *  USED FOR FLAM PUT
      *
          02  DATLEN    PIC S9(8)  COMP SYNC VALUE +80.
          02  DATABYTES PIC X(80).
      /
       PROCEDURE DIVISION.
       MAIN SECTION.
      *
       OPEN-INPUT-DATA.
      *
      *  OPEN DATA SET TO READ RECORDS
      *
          OPEN INPUT INDAT.
       OPEN-FLAM.
      *
      *    OPEN FLAM FOR OUTPUT (COMPRESSION)
      *
          CALL  "FLMOPN" USING FILE-ID, RETCO,
                              LASTPAR, OPENMODE, DDNAME, STATIS.
          IF  NOT FLAMOK
             THEN  MOVE "OPEN" TO OPERATION
                   PERFORM FLAM-ERROR
                   GO TO CLOSE-DATA.

       READ-RECORD.
      *
      *  READ A RECORD FROM INPUT DATA SET
      *
          READ INDAT INTO DATABYTES  AT END
                                     GO TO FINISH-COMPRESSION.
      *
       WRITE-RECORD.
      *
      *  WRITE THE RECORD WITH FLAM COMPRESSION
      *
          CALL "FLMPUT" USING FILE-ID, RETCO,
                             DATLEN, DATABYTES.
      *
```

```
     IF  FLAMOK
       THEN  GO TO READ-RECORD
       ELSE  MOVE "PUT" TO  OPERATION
             PERFORM FLAM-ERROR.
*
 FINISH-COMPRESSION.
*
*  CLOSE FLAM
*
     CALL "FLMCLS" USING FILE-ID, RETCO.
     IF  NOT FLAMOK
       THEN MOVE "CLOSE" TO OPERATION
            PERFORM FLAM-ERROR.
 CLOSE-DATA.
     CLOSE  INDAT.
 MAIN-END.
     STOP RUN.
*
 FLAM-ERROR SECTION.
 FLAM-ERROR-1.
     IF   NODMS-ERROR
       THEN   DISPLAY "FLAM-ERROR." UPON OUT-PUT
       ELSE   MOVE LOW-VALUE TO RETCO-1
              DISPLAY "DMS-ERROR FOR FLAMFILE." UPON OUT-PUT.
     DISPLAY "OPERATION " OPERATION "RETURNCODE= " RETCO
             UPON OUT-PUT.
 FLAM-ERROR-99.
     EXIT.
```

5.2.2 Dekomprimieren

Hier liest FLAM das Komprimat aus dem vorangegangenen Beispiel. ber die Satzschnittstelle werden die dekomprimierten S tze bereitgestellt und mit COBOL in die sequentielle Datei "OUTDAT" geschrieben.

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.  SAMPLE1D.
 AUTHOR.      LIMES DATENTECHNIK GMBH.
*
*  SAMPLE1D READS WITH FLAM COMPRESSED RECORDS AND WRITES
*           THE RECEIVED DECOMPRESSED DATA IN A SEQUENTIAL
*           DATA SET.
*
*           IN THIS EXAMPLE, THE FLAMFILE CAN BE
*              - ANY DATA SET     IN  MVS, BS2000
*              - VSAM             IN VSE/ESA
*
*           HIER WIRD MIT FLAM AUF KOMPRIMIERTE DATEN LESEND
*           ZUGEGRIFFEN.
*           DIE ERHALTENEN DATENSAETZE WERDEN IN EINE SEQUENT.
*           DATEI GESCHRIEBEN.
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
*
 SPECIAL-NAMES.
     SYSOUT IS  OUT-PUT.
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
     SELECT      OUTDAT
     ASSIGN TO SYS010-S-DATAOUT
     ACCESS MODE IS SEQUENTIAL.
*
*       DATA DIVISION.
*
 FILE SECTION.
 FD  OUTDAT  RECORD CONTAINS 80 CHARACTERS
             RECORDING MODE F.
 01  OUTDAT-RECORD.
     02 FILLER   PIC X(80).

*
 WORKING-STORAGE SECTION.
*
 77  OPERATION   PIC  X(6).
*
 01  FLAM-PARAMETER.
*
*  USED FOR ALL FLAM-CALLS
*
     02 FILE-ID  PIC S9(8)  COMP SYNC.
     02 RETCO    PIC S9(8)  COMP SYNC.
        88 FLAMOK            VALUE  0.
        88 FILEID-ERR        VALUE -1.
        88 MEMORY-ERR        VALUE -1.
```



```
          88 REC-TRUNCATED      VALUE  1.
          88 END-OF-FILE        VALUE  2.
          88 REC-NOT-FOUND      VALUE  5.
          88 NEW-HEADER         VALUE  6.
*
          88 NO-FLAMFILE        VALUE 10.
          88 FORMAT-ERR         VALUE 11.
          88 RECLEN-ERR         VALUE 12.
          88 FILELEN-ERR        VALUE 13.
          88 CHECKSUM-ERR       VALUE 14.
          88 MAXB-INVALID       VALUE 21.
          88 COMPMODE-INVALID   VALUE 22.
          88 COMPSYNTAX-ERR     VALUE 23.
          88 MAXREC-INVALID     VALUE 24.
          88 MAXSIZE-INVALID    VALUE 25.
          88 FLAMCODE-INVALID   VALUE 26.
          88 FILE-EMPTY         VALUE 30.
          88 NO-DATA-SET        VALUE 31.
*
       02  RETCO-X   REDEFINES RETCO.
         03 RETCO-1   PIC X
           88 FLAM-ERROR-RC VALUE  LOW-VALUE.
         03 RETCO-2-4 PIC XXX.
*
*  USED FOR FLAM OPEN
*
       02  LASTPAR   PIC S9(8)  COMP SYNC VALUE 0.
       02  OPENMODE  PIC S9(8)  COMP SYNC VALUE 0.
       02  DDNAME    PIC X(8)   VALUE "FLAMFIL".
       02  STATIS    PIC S9(8)  COMP SYNC VALUE 0.
*
*  USED FOR FLAM GET
*
       02  DATLEN    PIC S9(8).
       02  MAXLEN    PIC S9(8)  COMP SYNC VALUE +80.

/
 PROCEDURE DIVISION.
*
 MAIN SECTION.
*
 OPEN-OUTPUT-DATA.
*
*  OPEN DATA SET TO WRITE RECORDS
*
       OPEN OUTPUT OUTDAT.
*
 OPEN-FLAM.
*
*  OPEN FLAM FOR INPUT (DECOMPRESSION)
*
       CALL  "FLMOPN" USING FILE-ID, RETCO,
                            LASTPAR, OPENMODE, DDNAME, STATIS.
       IF  NOT FLAMOK
          THEN  MOVE "OPEN" TO OPERATION
                PERFORM FLAM-ERROR
                GO TO CLOSE-DATA.
```

```
 READ-RECORD.
*
*  READ A RECORD WITH FLAM IN OUTPUT AREA
*
     CALL "FLMGET" USING FILE-ID, RETCO,
                         DATLEN, OUTDAT-RECORD, MAXLEN.
*
     IF  FLAMOK
       THEN  NEXT SENTENCE
       ELSE  IF  END-OF-FILE
                THEN GO TO CLOSE-FLAM
                ELSE MOVE "GET" TO OPERATION
                     PERFORM FLAM-ERROR
                     GO TO CLOSE-FLAM.
*
 WRITE-RECORD.
*
*  WRITE THE DECOMPRESSED RECORD
*
     WRITE OUTDAT-RECORD.
*
     GO TO READ-RECORD.
*

 CLOSE-FLAM.
*
*  CLOSE TO FLAM
*
     CALL "FLMCLS" USING FILE-ID, RETCO.
     IF  NOT FLAMOK
       THEN MOVE "CLOSE" TO OPERATION
            PERFORM FLAM-ERROR.
 CLOSE-DATA.
*
*  CLOSE OUTPUT DATA
*
     CLOSE  OUTDAT.
 MAIN-END.
     STOP RUN.
*
 FLAM-ERROR SECTION.
 FLAM-ERROR-1.
     IF   FLAM-ERROR-RC
       THEN   DISPLAY "FLAM-ERROR." UPON OUT-PUT
       ELSE   MOVE LOW-VALUE TO RETCO-1
              DISPLAY "DMS-ERROR FOR FLAMFILE." UPON OUT-PUT.
     DISPLAY "OPERATION " OPERATION "RETURNCODE= " RETCO
             UPON OUT-PUT.
 FLAM-ERROR-99.
     EXIT.
```


5.2.3 Direktzugriff auf indexsequentielle FLAMFILE

Dieses Beispiel setzt als Eingabe eine indexsequentielle FLAMFILE einer indexsequentiellen Originaldatei mit 80 Bytes Satzl nge und Satzschl sseln von 8 Bytes L nge an der Position 73 voraus. Die Schl ssel sind abdruckbar numerisch von 1 bis n, wobei n gr er als 40 sein sollte. Das Komprimat dieser Datei kann mit dem Dienstprogramm FLAM erzeugt werden.

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.  SAMPLE3D.
 AUTHOR.      LIMES DATENTECHNIK GMBH.
*
*  SAMPLE3D IS AN EXAMPLE FOR AN INFORMATION RETRIEVAL PROGRAM,
*  BASED ON A VSAM-KSDS-FLAMFILE, USING THE FLAM-CALL-INTERFACE
*
*  A DIRECT READ WITH KEY IS DONE.
*  IF RECORD FOUND, THE NEXT RECORDS ARE READ SEQUENTIAL AND
*  DISPLAYED, UNTIL A NEW SET OF KEYS START.
*
 ENVIRONMENT DIVISION.
*
 CONFIGURATION SECTION.
*
 SPECIAL-NAMES.
*
     SYSOUT IS OUT-PUT.
*
 DATA DIVISION.
*
 WORKING-STORAGE SECTION.
*
 77  NEXT-KEY        PIC  9(8).
*
 77  CONDITION-FLAG  PIC  X.
     88 SET-END      VALUE "X".
*
 77  SET-END-FLAG    PIC  X     VALUE "X".
*
 01  FLAM-FILEID     PIC  9(8)  COMP.
*
 01  FLAM-RETCO      PIC S9(8)  COMP.
     88 FLAMOK            VALUE  0.
     88 FILEID-ERR        VALUE -1.
     88 MEMORY-ERR        VALUE -1.
     88 REC-TRUNCATED     VALUE  1.
     88 END-OF-FILE       VALUE  2.
     88 REC-NOT-FOUND     VALUE  5.
     88 NEW-HEADER        VALUE  6.
*

     88 NO-FLAMFILE       VALUE 10.
     88 FORMAT-ERR        VALUE 11.
     88 RECLEN-ERR        VALUE 12.
     88 FILELEN-ERR       VALUE 13.
```

```
     88 CHECKSUM-ERR      VALUE 14.
     88 MAXB-INVALID      VALUE 21.
     88 COMPMODE-INVALID  VALUE 22.
     88 COMPSYNTAX-ERR    VALUE 23.
     88 MAXREC-INVALID    VALUE 24.
     88 MAXSIZE-INVALID   VALUE 25.
     88 FLAMCODE-INVALID  VALUE 26.
     88 FILE-EMPTY        VALUE 30.
*
 01  RETCO-X REDEFINES FLAM-RETCO.
     03 RETCO-1   PIC X.
        88 NODMS-ERROR    VALUE  LOW-VALUE.
     03 RETCO-2   PIC X.
     03 RETCO-3-4.
      05 RETCO-3  PIC X.
      05 RETCO-4  PIC X.
****************************************************
*
 01  FLMOPN-AREA.
     02  LASTPAR   PIC S9(8)  COMP SYNC VALUE 0.
     02  OPENMODE  PIC S9(8)  COMP SYNC VALUE 0.
     02  DDNAME    PIC X(8)   VALUE "FLAMFIL".
     02  STATIS    PIC S9(8)  COMP SYNC VALUE 0.
*
 01  FLMGET-FLMGKY-AREA.
     02 DATALEN      PIC S9(8)  COMP SYNC.
     02 DATA-AREA.
        04 PURE-DATA PIC X(72).
        04 KEY-DATA  PIC 9(8).
     02 BUFFLEN      PIC S9(8)  COMP SYNC VALUE +80.
*
 01  SEARCH-KEYS.
     02 S-KEY-1      PIC  9(8)  VALUE  10.
     02 S-KEY-2      PIC  9(8)  VALUE  30.
     02 S-KEY-3      PIC  9(8)  VALUE  0.
 01  STOP-KEYS.
     02 STOP-KEY-1   PIC  9(8)  VALUE  20.
     02 STOP-KEY-2   PIC  9(8)  VALUE  40.
     02 STOP-KEY-3   PIC  9(8)  VALUE  9.
/
 PROCEDURE DIVISION.
*
 MAIN SECTION.
 MAIN-OPEN-FILE.
*
*    OPEN FLAMFILE
*
*    THE FLAMFILE WAS BUILD BY THE FLAM-UTILITY, SO IT HAS
*    A FILE-HEADER WITH VALUES ABOUT THE ORIGINAL DATA SET.
*    THEN WE NEED ONLY THE FLMOPN-CALL.
*
     CALL "FLMOPN" USING  FLAM-FILEID,
                          FLAM-RETCO,
                          LASTPAR,
                          OPENMODE,
                          DDNAME,
                          STATIS.
```

```
          IF   NOT FLAMOK
             THEN  DISPLAY "OPEN-ERROR." UPON OUT-PUT
                   PERFORM FLAM-ERROR
                   GO TO MAIN-END.
      MAIN-SEARCH-1.
     *
     *  SEARCH FOR SPECIAL RECORD WITH KEY NO. 1
     *
          MOVE   S-KEY-1        TO    KEY-DATA.
          PERFORM  GET-KEY.
     *
     *  IF RECORD FOUND, READ THE NEXT RECORDS
     *
          IF   FLAMOK
             THEN MOVE   STOP-KEY-1  TO  NEXT-KEY
                  MOVE   SPACE       TO  CONDITION-FLAG
                  PERFORM GET-SEQ UNTIL SET-END.
      MAIN-SEARCH-2.
     *
     *  SEARCH FOR SPECIAL RECORD WITH KEY NO. 2
     *
          MOVE   S-KEY-2        TO    KEY-DATA.
          PERFORM  GET-KEY.
     *
     *  IF RECORD FOUND, READ THE NEXT RECORDS
     *
          IF   FLAMOK
             THEN MOVE   STOP-KEY-2  TO  NEXT-KEY
                  MOVE   SPACE       TO  CONDITION-FLAG
                  PERFORM GET-SEQ UNTIL SET-END.

      MAIN-SEARCH-3.
     *
     *  SEARCH FOR SPECIAL RECORD WITH KEY NO. 3
     *  (KEY DOES NOT EXIST IN DATA SET).
     *
          MOVE   S-KEY-3        TO    KEY-DATA.
          PERFORM  GET-KEY.
     *
     *  IF RECORD NOT FOUND, FLAM POSITIONS TO THE NEXT HIGHER KEY
     *  IN THE DATA SET:
     *
          IF   REC-NOT-FOUND
             THEN MOVE   STOP-KEY-3  TO  NEXT-KEY
                  MOVE   SPACE       TO  CONDITION-FLAG
                  PERFORM GET-SEQ UNTIL SET-END.
      MAIN-CLOSE-FILE.
     *
     *  CLOSE FLAMFILE
     *
           CALL "FLMCLS" USING  FLAM-FILEID,
                                      FLAM-RETCO.
      MAIN-END.
          STOP RUN.
     /
      FLAM-ERROR SECTION.
     *
```

```
*   FLAM-RETURNCODE IS NOT ZERO.
*   DOCUMENT THE ERROR-SITUATION.
*
 FLAM-ERROR-1.
     IF   END-OF-FILE
       THEN   GO TO FLAM-ERROR-99.
     IF   NODMS-ERROR
       THEN   DISPLAY "FLAM-ERROR." UPON OUT-PUT
       ELSE   MOVE LOW-VALUE TO RETCO-1
*             THIS BYTE CONTAINS A SIGN FOR DATA SET-ERROR,
*             WE DON'T NEED TO DISPLAY IT
              DISPLAY "DMS-ERROR FOR FLAMFILE." UPON OUT-PUT.
 FLAM-ERROR-2.
     DISPLAY "RETURNCODE= " FLAM-RETCO UPON OUT-PUT.
 FLAM-ERROR-99.
     EXIT.
/
 GET-KEY SECTION.
*
*   GET A RECORD WITH SPECIFIED KEY
*
 GET-KEY-1.
     CALL "FLMGKY" USING   FLAM-FILEID,
                           FLAM-RETCO,
                           DATALEN,
                           DATA-AREA,
                           BUFFLEN.
 GET-KEY-2.
     IF  FLAMOK
        THEN  NEXT SENTENCE
        ELSE  IF   REC-NOT-FOUND
                 THEN  DISPLAY "KEY NOT FOUND:    "  KEY-DATA
                               UPON OUT-PUT
                       GO TO GET-KEY-99
                 ELSE  PERFORM FLAM-ERROR
                       GO TO GET-KEY-99.
 GET-KEY-3.
     DISPLAY "KEY FOUND: " KEY-DATA UPON OUT-PUT.
     DISPLAY "DATA: "               UPON OUT-PUT.
     DISPLAY DATA-AREA              UPON OUT-PUT.
 GET-KEY-99.
     EXIT.
/
 GET-SEQ SECTION.
*
*   GET RECORDS IN SEQUENTIAL ORDER
*
 GET-SEQ-1.
     CALL "FLMGET" USING   FLAM-FILEID,
                           FLAM-RETCO,
                           DATALEN,
                           DATA-AREA,
                           BUFFLEN.
 GET-SEQ-2.
*
*   CHECK RETURNCODE
*
```

```
     IF   FLAMOK
       THEN
*
*      IF RECORD CONTAINS TO THE SET, DISPLAY THE DATA,
*      ELSE SET THE SET-END CONDITION.
*
         IF  KEY-DATA  < NEXT-KEY
           THEN  DISPLAY  DATA-AREA  UPON OUT-PUT
           ELSE  MOVE SET-END-FLAG TO CONDITION-FLAG
       ELSE
*
*      SET THE SET-END CONDITION,
*      ON ERROR, DISPLAY THE FLAM-RETURNCODE.
*
           MOVE  SET-END-FLAG  TO    CONDITION-FLAG
           IF   NOT END-OF-FILE
             THEN  PERFORM FLAM-ERROR.
 GET-SEQ-99.
     EXIT.
```

5.2.4 Testprogramm f r die Satzschnittstelle FLAMREC

Mit diesem Programm k nnen Funktionen der Satzschnittstelle FLAMREC mit allen Parameterwerten in beliebiger Reihenfolge aufgerufen werden. Dieses Beispiel enth lt damit alle Datendefinitionen und alle Unterprogrammaufrufe, die f r die Satzschnittstelle gebraucht werden k nnen. Es kann sowohl als Muster f r eigene Entwicklungen als auch zum Untersuchen beliebiger Komprimatsdateien verwendet werden.

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID. RECTEST.
*****************************************************************
*  NAME:     RECTEST                                            *
*                                                               *
*  FUNKTION: FLAMREC-SCHNITTSTELLE BENUTZEN.                    *
*            MIT DIESEM TESTPROGRAMM KOENNEN ALLE FUNKTIONEN    *
*            DER FLAM SATZSCHNITTSTELLE FLAMREC MIT ALLEN PARA- *
*            METERWERTEN IN BELIEBIGER REIHENFOLGE AUFGERUFEN   *
*            WERDEN.                                            *
*                                                               *
*  FUNCTION: USE THE INTERFACE FLAMREC.                         *
*            IN THIS EXAMPLE YOU ARE ABLE TO CALL ALL FUNCTIONS *
*            OF THE RECORD INTERFACE FLAMREC WITH ALL PARAMETERS*
*            AND IN EVERY SEQUENCE.                             *
*                                                               *
*                                                               *
*                                                               *
*  IT IS AN EXAMPLE FOR CALLING FLAM FROM A COBOL ROUTINE.      *
*  EVERY ENTRY AND ITS PARAMETER ARE DESCRIBED.                 *
*                                                               *
*  TO START RECTEST IN TSO, USE AFTER COMPILATION AND LINKING:  *
*                                                               *
*    ALLOC DSN(*) DD(SYSIN)                                     *
*    ALLOC DSN(*) DD(SYSOUT)                                    *
*    CALL FLAMV27C(RECTEST)                                     *
*                                                               *
*****************************************************************
 ENVIRONMENT DIVISION. CONFIGURATION SECTION.
 SPECIAL-NAMES.
*
     SYSIN    IS TERMIN
     SYSOUT   IS TERMOUT.
*
 DATA DIVISION.
 WORKING-STORAGE SECTION.
*
*    PARAMETER FOR FLMOPN
*

 77  FLAMID                     PIC S9(8) COMP SYNC.
 01  RETCO                      PIC S9(8) COMP SYNC.
     88  OK                         VALUE 0.
     88  UNZULAESSIG                VALUE -1.
```



```
 01  RETCO-RED REDEFINES RETCO.
     05  RETCO-INDICATOR          PIC X(1).
         88  DVS-ERROR                VALUE HIGH-VALUE.
     05  FILLER                   PIC X(1).
     05  RETCO-FLAM               PIC S9(4) COMP SYNC.
         88  CUT                      VALUE 1.
         88  EOF                      VALUE 2.
         88  GAP                      VALUE 3.
         88  INVKEY                   VALUE 5.
 77  LASTPAR                      PIC S9(8) COMP SYNC
                                      VALUE 1.
     88  LAST-PARAMETER               VALUE 0.
 77  OPENMODE                     PIC S9(8) COMP SYNC
                                      VALUE 2.
     88  OPEN-INPUT                   VALUE 0.
     88  OPEN-OUTPUT                  VALUE 1.
     88  OPEN-INOUT                   VALUE 2.
     88  OPEN-OUTIN                   VALUE 3.
 77  DDNAME                       PIC X(8)
                                      VALUE "FLAMFIL".
 77  STATIS                       PIC S9(8) COMP SYNC
                                      VALUE 1.
     88  STATISTIK                    VALUE 1.
*
*    PARAMETER FOR FLMOPD
*
 77  NAMELEN                      PIC S9(8) COMP SYNC
                                      VALUE 54.
 77  FILENAME                     PIC X(54)
                                      VALUE SPACES.
 77  DSORG                        PIC S9(8) COMP SYNC
                                      VALUE 1.
 77  RECFORM                      PIC S9(8) COMP SYNC.
 77  MAXSIZE                      PIC S9(8) COMP SYNC
                                      VALUE 512.
 77  RECDELIM                     PIC X(4).
 77  BLKSIZE                      PIC S9(8) COMP SYNC.
 77  CLOSDISP                     PIC S9(8) COMP SYNC
                                      VALUE 0.
 77  DEVICE                       PIC S9(8) COMP SYNC
                                      VALUE 0.
*
*    PARAMETER FOR FLMOPF
*
 77  VERSION                      PIC S9(8) COMP SYNC.
     88  VERSION-1                    VALUE 100.
     88  VERSION-1-1                  VALUE 101.
     88  VERSION-2                    VALUE 200.
 77  FLAMCODE                     PIC S9(8) COMP SYNC.
     88  EBC-DIC                      VALUE 0.
     88  ASCII                        VALUE 1.

 77  COMPMODE                     PIC S9(8) COMP SYNC.
     88  CX8                          VALUE 0.
     88  CX7                          VALUE 1.
     88  VR8                          VALUE 2.
 77  MAXBUFF                      PIC S9(8) COMP SYNC.
```

```
 77  HEADER                     PIC S9(8) COMP SYNC
                                    VALUE 1.
     88  NOHEADER                   VALUE 0.
     88  FILEHEADER                 VALUE 1.
 77  MAXREC                     PIC S9(8) COMP SYNC
                                    VALUE 255.
*
*    SCHLUESSELBESCHREIBUNG DER FLAMFILE
*    KEY DESCRIPTION OF THE FLAMFILE
*
 01  KEYDESC.
   05  KEYFLAGS                 PIC S9(8) COMP SYNC
                                    VALUE 1.
   05  KEYPARTS                 PIC S9(8) COMP SYNC
                                    VALUE 1.
   05  KEYENTRY1.
     10  KEYPOS1                PIC S9(8) COMP SYNC
                                    VALUE 1.
     10  KEYLEN1                PIC S9(8) COMP SYNC
                                    VALUE 9.
     10  KEYTYPE1               PIC S9(8) COMP SYNC
                                    VALUE 1.
   05  KEYENTRY-2-BIS-8             OCCURS 7 TIMES.
     10  KEYPOS                 PIC S9(8) COMP SYNC.
     10  KEYLEN                 PIC S9(8) COMP SYNC.
     10  KEYTYPE                PIC S9(8) COMP SYNC.
*
 77  BLKMODE                    PIC S9(8) COMP SYNC.
     88  UNBLOCKED                  VALUE 0.
     88  BLOCKED                    VALUE 1.
 77  EXK20                      PIC X(8)
                                    VALUE SPACES.
 77  EXD20                      PIC X(8)
                                    VALUE SPACES.
*
*    PARAMETER FOR FLMPHD
*
 77  NAMELEN-ORIG               PIC S9(8) COMP SYNC
                                    VALUE 54.
 77  FILENAME-ORIG              PIC X(54)
                                    VALUE SPACES.
 77  DSORG-ORIG                 PIC S9(8) COMP SYNC
                                    VALUE 1.
 77  RECFORM-ORIG               PIC S9(8) COMP SYNC.
 77  RECSIZE-ORIG               PIC S9(8) COMP SYNC
                                    VALUE 512.
 77  RECDELIM-ORIG              PIC X(4).
 77  BLKSIZE-ORIG               PIC S9(8) COMP SYNC.
 77  PRCTRL-ORIG                PIC S9(8) COMP SYNC
                                    VALUE 0.

     88  NO-CONTROL-CHAR            VALUE 0.
     88  ASA-CONTROL-CHAR           VALUE 1.
     88  MACH-CONTROL-CHAR          VALUE 2.
 77  SYSTEM-ORIG                PIC X(2)
                                    VALUE LOW-VALUES.
 77  LASTPAR-PHD                PIC S9(8) COMP SYNC
```

```
                                    VALUE 1.
     88  LAST-PARAMETER-PHD         VALUE 0.
*
*    SCHLUESSELBESCHREIBUNG DER ORIGINALDATEI
*    KEY DESCRIPTION OF THE ORIGINAL DATA SET
*
 01  KEYDESC-ORIG.
   05  KEYFLAGS-ORIG            PIC S9(8) COMP SYNC
                                    VALUE 1.
   05  KEYPARTS-ORIG            PIC S9(8) COMP SYNC
                                    VALUE 1.
   05  KEYENTRY1-ORIG.
     10  KEYPOS1-ORIG           PIC S9(8) COMP SYNC
                                    VALUE 1.
     10  KEYLEN1-ORIG           PIC S9(8) COMP SYNC
                                    VALUE 8.
     10  KEYTYPE1-ORIG          PIC S9(8) COMP SYNC
                                    VALUE 1.
   05  KEYENTRY-2-BIS-8-ORIG        OCCURS 7 TIMES
                                    INDEXED BY KEYDESC-INDEX.
     10  KEYPOS-ORIG            PIC S9(8) COMP SYNC.
     10  KEYLEN-ORIG            PIC S9(8) COMP SYNC.
     10  KEYTYPE-ORIG           PIC S9(8) COMP SYNC.
*
 77  KEYDESC-INDIKATOR          PIC X(1)
                                    VALUE "Y".
     88  KEYDESC-DEFINIERT          VALUE "Y".
*
*    PARAMETER FOR FLMPUH
*
 77  UATTRLEN                   PIC S9(8) COMP SYNC.
 77  USERATTR                   PIC X(80).
*
*    PARAMETER FLMGET / FLMPUT
*
 77  RECLEN                     PIC S9(8) COMP SYNC
                                          VALUE 80.
 01  REC-ORD.
     05  BYTE                   PIC X(1)
                                    OCCURS  32767 TIMES
                                    INDEXED BY REC-INDEX.
 01  RECORD-DISPLAY REDEFINES REC-ORD
                                PIC X(80).
 01  RECORD-KEY-DISPLAY.
     02 RECORD-KEY-BYTE         PIC X(1)   OCCURS 80
                                    INDEXED BY KEY-INDEX.
 77  BUFLEN                     PIC S9(8) COMP SYNC
                                    VALUE 32767.
 77  CHECKMODE                  PIC S9(8) COMP SYNC
                                    VALUE 0.

 77  RECNO                      PIC S9(8) COMP SYNC
                                    VALUE 0.
*
*    VARIABLEN ZUR AUFBEREITUNG DES RETURNCODES
*
 77  LEN-RETCO                  PIC S9(8) COMP SYNC
```

```
                                        VALUE 4.
 01  RETCO-HEX.
     05  FILLER                 PIC X(4).
     05  RETCO-DISP             PIC X(4).
*
*    VARIABLEN ZUM EINLESEN UND AUFBEREITEN VON ZAHLEN
*    VARIABLES TO READ IN, AND EDIT NUMBERS
*
 01  EINGABE.
     05  BYTE-EIN               PIC X(1)
                                OCCURS 9 TIMES
                                INDEXED BY EIN-INDEX.
 01  EINGABE-NUM                PIC S9(8).
 01  EINGABE-RED REDEFINES EINGABE-NUM.
     05  BYTE-RED               PIC X(1)
                                OCCURS 8 TIMES
                                INDEXED BY RED-INDEX.
*
*  AUSGEWAEHLTE FUNKTION
*  SELECTED FUNCTION
*
 01  FUNKTION                   PIC X(8).
     88  FLMOPD                    VALUES "FLMOPD" "OPD".
     88  FLMOPF                    VALUES "FLMOPF" "OPF".
     88  FLMCLS                    VALUES "FLMCLS" "CLS" "C".
     88  FLMFLU                    VALUES "FLMFLU" "FLU".
     88  FLMGET                    VALUES "FLMGET" "GET" "G".
     88  FLMGTR                    VALUES "FLMGTR" "GTR".
     88  FLMGKY                    VALUES "FLMGKY" "GKY".
     88  FLMFKY                    VALUES "FLMFKY" "FKY".
     88  FLMGRN                    VALUES "FLMGRN" "GRN".
     88  FLMFRN                    VALUES "FLMFRN" "FRN".
     88  FLMPUT                    VALUES "FLMPUT" "PUT" "P".
     88  FLMPKY                    VALUES "FLMPKY" "PKY".
     88  FLMPOS                    VALUES "FLMPOS" "POS".
     88  FLMDEL                    VALUES "FLMDEL" "DEL" "D".
     88  FLMUPD                    VALUES "FLMUPD" "UPD" "U".
     88  FLMPHD                    VALUES "FLMPHD" "PHD".
     88  FLMPUH                    VALUES "FLMPUH" "PUH".
     88  FLMGHD                    VALUES "FLMGHD" "GHD".
     88  FLMGUH                    VALUES "FLMGUH" "GUH".

*
*  AREAS FOR FLMCLS AND FLMFLU
*
 77  CPUTIME                    PIC 9(8) COMP.
 77  REC-ORDS                   PIC 9(8) COMP.
 01  BYTEFELD.
     05  BYTEOFL                PIC 9(8) COMP SYNC.
     05  BYTES                  PIC 9(8) COMP SYNC.
 01  BYTECNT REDEFINES BYTEFELD PIC S9(18) COMP SYNC.
 77  CMPRECS                    PIC 9(8) COMP.
 01  CMPBYFELD.
     05  CMPBYOFL               PIC 9(8) COMP SYNC.
     05  CMPBYTES               PIC 9(8) COMP SYNC.
 01  CMPBYCNT REDEFINES CMPBYFELD
                                PIC S9(18) COMP SYNC.
```

```
*
 77  STATIS-DIS                 PIC ZZZ,ZZZ,ZZZ,ZZZ,ZZZ,ZZ9.
*
*  ARBEITSVARIABLEN
*  WORK FIELDS
*
 77  INDEX-DISPLAY              PIC 9(8).
 77  KEY-IND-DISP               PIC S9(8) COMP.
 77  GET-COUNT                  PIC 9(8).
 77  GET-INDEX                  PIC S9(8) COMP SYNC.
 77  REL-POSITION               PIC S9(8) COMP SYNC.
     88  DATEI-ENDE                 VALUE  99999999.
     88  DATEI-ANFANG               VALUE -99999999.
 77  DIGIT                      PIC 9.
 01  HEXDATA                    PIC 9(8) COMP SYNC.
 01  HEXDATA-BYTES REDEFINES HEXDATA.
     05  BYTE-1-2-HEX           PIC X(2).
     05  BYTE-3-4-HEX           PIC X(2).
 77  HEX-QUOTIENT               PIC 9(8) COMP SYNC.
 77  HEX-REMAINDER              PIC 9(8) COMP SYNC.
 01  HEXDIGITS                  PIC X(16)
                                    VALUE "0123456789ABCDEF".
 01  HEXTAB REDEFINES HEXDIGITS.
     05  DIGIT-HEX              PIC X(1)
                                    OCCURS 16 TIMES
                                    INDEXED BY HEX-INDEX.
 01  CHARDATA                   PIC X(8).
 01  CHARDATA-BYTES REDEFINES CHARDATA.
     05  BYTE-1-CHAR            PIC X(2).
     05  BYTE-2-4-CHAR.
         10  BYTE-2-CHAR        PIC X(2).
         10  BYTE-3-4-CHAR      PIC X(4).
 01  CHARDATA-TAB   REDEFINES CHARDATA.
     05  BYTE-CHAR              PIC X(1)
                                    OCCURS 8 TIMES
                                    INDEXED BY CHAR-INDEX.
/
 PROCEDURE DIVISION.
*
*  STARTMELDUNG AUSGEBEN
*
 START-MELDUNG.
*
     DISPLAY " "                                   UPON TERMOUT.
     DISPLAY "START PROGRAM TO TEST FLAMREC"       UPON TERMOUT.
     DISPLAY " "                                   UPON TERMOUT.
*
*  DATEI OEFFNEN
*
 OPEN-EINGABE.
*
     DISPLAY "ENTER PARAMETER FOR FLMOPN: "        UPON TERMOUT
     DISPLAY " "                                   UPON TERMOUT
     DISPLAY "OPENMODE (0=INPUT 1=OUTPUT 2=INOUT) ?"
                                                   UPON TERMOUT
     PERFORM  NUMERISCHE-EINGABE
     MOVE     EINGABE-NUM TO OPENMODE
```

```
     DISPLAY "DDNAME ?"                               UPON TERMOUT
     ACCEPT   DDNAME                                  FROM TERMIN
     DISPLAY "STATIS (0=NO 1=YES) ?"                  UPON TERMOUT
     PERFORM  NUMERISCHE-EINGABE
     MOVE     EINGABE-NUM TO STATIS
     DISPLAY "LASTPAR (0=YES 1=NO) ?"                 UPON TERMOUT
     PERFORM  NUMERISCHE-EINGABE
     MOVE     EINGABE-NUM TO LASTPAR
*
     CALL    "FLMOPN" USING FLAMID, RETCO,
                            LASTPAR, OPENMODE,
                            DDNAME, STATIS
     IF   NOT OK
     THEN
          DISPLAY "ERROR ON 'OPEN' OF: ", DDNAME UPON TERMOUT
          PERFORM  FEHLER-MELDUNG
          DISPLAY " "                                 UPON TERMOUT
          DISPLAY "PROGRAM TERMINATED WITH ERRORS" UPON TERMOUT
          STOP RUN
     END-IF.
*
 OPEN-NEXT.
*
     IF   NOT LAST-PARAMETER
     THEN
          DISPLAY "PLEASE SELECT FUNCTION: FLMOPD FLMOPF"
                                                      UPON TERMOUT
          ACCEPT   FUNKTION                           FROM TERMIN
          IF   FLMOPD
          THEN
               DISPLAY " "                            UPON TERMOUT
               DISPLAY "ENTER PARAMETER FOR FLMOPD: "

                                                      UPON TERMOUT
               DISPLAY "DATA SET NAME ?"              UPON TERMOUT
               ACCEPT   FILENAME                      FROM TERMIN
               DISPLAY "NAMELEN (0 - 54) ?"           UPON TERMOUT
               PERFORM  NUMERISCHE-EINGABE
               MOVE     EINGABE-NUM TO NAMELEN
               IF   OPEN-OUTPUT
               THEN
                    DISPLAY "DSORG   (0=SEQ 1=INDEX ...) ?"
                                                      UPON TERMOUT
                    PERFORM  NUMERISCHE-EINGABE
                    MOVE     EINGABE-NUM TO DSORG
                    DISPLAY "RECFORM (0=VAR 1=FIX ...) ?"
                                                      UPON TERMOUT
                    PERFORM  NUMERISCHE-EINGABE
                    MOVE     EINGABE-NUM TO RECFORM
                    DISPLAY "MAXSIZE (80 - 32768) ?"
                                                      UPON TERMOUT
                    PERFORM  NUMERISCHE-EINGABE
                    MOVE     EINGABE-NUM TO MAXSIZE
                    DISPLAY "KEYDESC FOR ORIGINAL DATA SET ?"
                                                      UPON TERMOUT
                    PERFORM  KEYDESC-EINGABE
                    MOVE     KEYDESC-ORIG TO KEYDESC
```

```
          DISPLAY "BLKSIZE (0 - 32768) ?"
                                         UPON TERMOUT
          PERFORM  NUMERISCHE-EINGABE
          MOVE     EINGABE-NUM TO BLKSIZE
     END-IF
     DISPLAY "CLOSDISP (0=REWIND 1=UNLOAD ...) ?"
                                         UPON TERMOUT
     PERFORM  NUMERISCHE-EINGABE
     MOVE     EINGABE-NUM TO CLOSDISP
     DISPLAY "DEVICE (0=DISK 1=TAPE ...) ?"
                                         UPON TERMOUT
     PERFORM  NUMERISCHE-EINGABE
     MOVE     EINGABE-NUM TO DEVICE
     DISPLAY "LASTPAR (0=YES 1=NO) ?"    UPON TERMOUT
     PERFORM  NUMERISCHE-EINGABE
     MOVE     EINGABE-NUM TO LASTPAR
     CALL    "FLMOPD" USING FLAMID, RETCO,
                      LASTPAR, NAMELEN, FILENAME,
                      DSORG, RECFORM, MAXSIZE,
                      RECDELIM, KEYDESC, BLKSIZE,
                      CLOSDISP, DEVICE
     IF   NOT OK
     THEN
          DISPLAY "ERROR ON 'OPEN' OF: ",
                   FILENAME              UPON TERMOUT
          PERFORM  FEHLER-MELDUNG
          DISPLAY " "                UPON TERMOUT
          DISPLAY "PROGRAM TERMINATED WITH ERRORS"
                                         UPON TERMOUT
          STOP RUN
     ELSE

          DISPLAY "NAMELEN  ", NAMELEN   UPON TERMOUT
          DISPLAY "DATA SET ", FILENAME  UPON TERMOUT
          DISPLAY "DSORG    ", DSORG     UPON TERMOUT
          DISPLAY "RECFORM  ", RECFORM   UPON TERMOUT
          DISPLAY "MAXSIZE  ", MAXSIZE   UPON TERMOUT
          IF   DSORG  0 AND KEYPARTS  0
          THEN
               DISPLAY "KEYDESC OF FLAMFILE"
                                         UPON TERMOUT
               DISPLAY "KEYFLAGS ", KEYFLAGS
                                         UPON TERMOUT
               DISPLAY "KEYPARTS ", KEYPARTS
                                         UPON TERMOUT
               DISPLAY "KEYPOS1  ", KEYPOS1
                                         UPON TERMOUT
               DISPLAY "KEYLEN1  ", KEYLEN1
                                         UPON TERMOUT
               DISPLAY "KEYTYPE1 ", KEYTYPE1
                                         UPON TERMOUT
          END-IF
          DISPLAY "BLKSIZE  ", BLKSIZE   UPON TERMOUT
          DISPLAY "CLOSDISP ", CLOSDISP  UPON TERMOUT
          DISPLAY "DEVICE   ", DEVICE    UPON TERMOUT
     END-IF
ELSE
```

```
                IF   FLMOPF
                THEN
                     MOVE     1        TO LASTPAR
                     MOVE     DDNAME TO FILENAME
                ELSE
                     DISPLAY  " UNKNOWN FUNCTION: ", FUNKTION
                                                    UPON TERMOUT
                     GO TO    OPEN-NEXT
                END-IF
           END-IF
 *
           IF   NOT LAST-PARAMETER
           THEN
                DISPLAY " "                         UPON TERMOUT
                DISPLAY "ENTER PARAMETER FOR FLMOPF :"
                                                    UPON TERMOUT
                IF   OPEN-OUTPUT
                THEN
                     DISPLAY "FLAMCODE (0=EBCDIC 1=ASCII) ?"
                                                    UPON TERMOUT
                     PERFORM  NUMERISCHE-EINGABE
                     MOVE     EINGABE-NUM TO FLAMCODE
                     DISPLAY "COMPMODE (0=CX8 1=CX7 2=VR8) ?"
                                                    UPON TERMOUT
                     PERFORM  NUMERISCHE-EINGABE
                     MOVE     EINGABE-NUM TO COMPMODE
                     DISPLAY "MAXBUFF (0 - 32768) ?"
                                                    UPON TERMOUT
                     PERFORM  NUMERISCHE-EINGABE
                     MOVE     EINGABE-NUM TO MAXBUFF
                     DISPLAY "HEADER (0=NO 1=YES) ?"
                                                    UPON TERMOUT
                     PERFORM  NUMERISCHE-EINGABE
                     MOVE     EINGABE-NUM TO HEADER
                     DISPLAY "MAXREC (1 - 255) ?"
                                                    UPON TERMOUT
                     PERFORM  NUMERISCHE-EINGABE
                     MOVE     EINGABE-NUM TO MAXREC
                     IF   FLMOPF
                     THEN
                          DISPLAY "KEYDESC FOR ORIGINALDATEI ?"
                                                    UPON TERMOUT
                          PERFORM  KEYDESC-EINGABE
                     END-IF
                     DISPLAY "BLKMODE (0=UNBLK 1=BLK) ?"
                                                    UPON TERMOUT
                     PERFORM  NUMERISCHE-EINGABE
                     MOVE     EINGABE-NUM TO BLKMODE
                     DISPLAY "EXK20 ?"              UPON TERMOUT
                     ACCEPT   EXK20                 FROM TERMIN
                ELSE
                     IF   FLMOPF
                     THEN
                          DISPLAY "HEADER (0=NO 1=YES) ?"
                                                    UPON TERMOUT
                          PERFORM  NUMERISCHE-EINGABE
                          MOVE     EINGABE-NUM TO HEADER
```



```
                         IF   OPEN-INOUT
                         THEN
                              DISPLAY "MAXREC (1 - 255) ?"
                                                          UPON TERMOUT
                              PERFORM  NUMERISCHE-EINGABE
                              MOVE     EINGABE-NUM TO MAXREC
                              DISPLAY "EXK20 ?"    UPON TERMOUT
                              ACCEPT   EXK20       FROM TERMIN
                         END-IF
                        DISPLAY "KEYDESC FOR ORIGINAL DATA SET ?"
                                                          UPON TERMOUT
                         PERFORM  KEYDESC-EINGABE
                    END-IF
                    DISPLAY "EXD20 ?"              UPON TERMOUT
                    ACCEPT   EXD20                 FROM TERMIN
               END-IF
               CALL    "FLMOPF" USING FLAMID, RETCO,
                                VERSION, FLAMCODE, COMPMODE,
                                MAXBUFF, HEADER, MAXREC,
                                KEYDESC-ORIG, BLKMODE,
                                EXK20, EXD20
*
               IF   NOT OK
               THEN
                    DISPLAY "ERROR ON OPEN OF: ",
                             FILENAME              UPON TERMOUT
                    PERFORM  FEHLER-MELDUNG
                    DISPLAY " "                    UPON TERMOUT
                    DISPLAY "PROGRAM TERMINATED WITH ERRORS"
                                                   UPON TERMOUT
                    STOP RUN
                ELSE
                    DISPLAY "VERSION  ", VERSION   UPON TERMOUT
                    DISPLAY "FLAMCODE ", FLAMCODE  UPON TERMOUT
                    DISPLAY "COMPMODE ", COMPMODE  UPON TERMOUT
                    DISPLAY "MAXBUFF  ", MAXBUFF   UPON TERMOUT
                    DISPLAY "HEADER   ", HEADER    UPON TERMOUT
                    DISPLAY "MAXREC   ", MAXREC    UPON TERMOUT
                    PERFORM  KEYDESC-AUSGABE
                    DISPLAY "BLKMODE  ", BLKMODE   UPON TERMOUT
                    DISPLAY "EXK20    ", EXK20     UPON TERMOUT
                    DISPLAY "EXD20    ", EXD20     UPON TERMOUT
                END-IF
           END-IF
     END-IF.
*
*****************************************************************
*  VERARBEITUNGSSCHLEIFE                                        *
*****************************************************************
*
     PERFORM UNTIL FLMCLS
        DISPLAY "PLEASE SELECT FUNCTION: "
                "GET GTR GKY FKY GRN FRN "
                "PUT PKY POS DEL UPD GHD GUH PHD PUH FLU CLS"
                                                   UPON TERMOUT
        ACCEPT  FUNKTION                           FROM TERMIN
        IF   FLMGET
```

```
THEN PERFORM SEQUENTIELL-LESEN
ELSE
 IF   FLMGTR
 THEN PERFORM SEQUENTIELL-LESEN-RUECKWAERTS
 ELSE
  IF   FLMPOS
  THEN PERFORM POSITIONIEREN
  ELSE
   IF   FLMDEL
   THEN PERFORM LOESCHEN
   ELSE
    IF   FLMGKY
    THEN PERFORM SCHLUESSEL-LESEN
    ELSE
     IF   FLMFKY
     THEN PERFORM SCHLUESSEL-POSITIONIEREN
     ELSE
      IF   FLMGRN
      THEN PERFORM SATZNUMMER-LESEN
      ELSE
       IF   FLMFRN
       THEN PERFORM SATZNUMMER-POSITIONIEREN
       ELSE
        IF   FLMPUT
        THEN PERFORM SCHREIBEN
        ELSE
         IF   FLMPKY
         THEN PERFORM SCHLUESSEL-SCHREIBEN
         ELSE
          IF   FLMUPD
          THEN PERFORM AENDERN
          ELSE
           IF   FLMPHD
           THEN PERFORM HEADER-SCHREIBEN
           ELSE
            IF   FLMPUH
            THEN PERFORM USER-HEADER-SCHREIBEN
            ELSE
             IF   FLMGHD
             THEN PERFORM HEADER-LESEN
             ELSE
              IF   FLMGUH
              THEN PERFORM USER-HEADER-LESEN
              ELSE
               IF   FLMFLU
               THEN PERFORM MATRIX-ABSCHLIESSEN
               ELSE
                IF   FLMCLS
                THEN DISPLAY FILENAME,
                             " WILL BE CLOSED"
                                          UPON TERMOUT
                ELSE DISPLAY FUNKTION,
                             " UNKNOWN"
                                          UPON TERMOUT
                END-IF
               END-IF
              END-IF
```

```
                      END-IF
                     END-IF
                    END-IF
                   END-IF
                  END-IF
                 END-IF
                END-IF
               END-IF
              END-IF
             END-IF
           END-IF
          END-IF
         END-IF
        END-IF
      END-PERFORM.
*
 FLAMFILE-SCHLIESSEN.
*
      CALL   "FLMCLS" USING FLAMID, RETCO CPUTIME REC-ORDS
                            BYTES BYTEOFL CMPRECS CMPBYTES
                            CMPBYOFL.
      IF NOT OK
         DISPLAY "ERROR DURING CLOSE"               UPON TERMOUT
         PERFORM FEHLER-MELDUNG
      ELSE
         IF STATISTIK
         THEN           DISPLAY " "                               UPON TERMOUT
            MOVE     REC-ORDS  TO STATIS-DIS
            DISPLAY "ORG.  RECORDS    ", STATIS-DIS UPON TERMOUT
            MOVE     BYTECNT   TO STATIS-DIS
            DISPLAY "ORG.  BYTES      ", STATIS-DIS UPON TERMOUT
            MOVE     CMPRECS   TO STATIS-DIS
            DISPLAY "COMP. RECORDS    ", STATIS-DIS UPON TERMOUT
            MOVE     CMPBYCNT  TO STATIS-DIS
            DISPLAY "COMP. BYTES      ", STATIS-DIS UPON TERMOUT
         END-IF
         DISPLAY " "                                UPON TERMOUT
         DISPLAY "PROGRAM NORMAL TERMINATION"       UPON TERMOUT
      END-IF.
      STOP RUN.
*
*****************************************************************
*  VERARBEITUNGSFUNKTIONEN                                      *
*  PROCESSING FUNCTIONS                                         *
*****************************************************************
*
 SEQUENTIELL-LESEN.
*
      DISPLAY "NUMBER RECORDS TO READ ?"            UPON TERMOUT.
      PERFORM NUMERISCHE-EINGABE.
      MOVE    EINGABE-NUM TO GET-COUNT.
      MOVE    0 TO RETCO.
      PERFORM VARYING GET-INDEX FROM 0 BY 1
              UNTIL GET-INDEX = GET-COUNT OR NOT OK
         MOVE    SPACES TO RECORD-DISPLAY
         CALL   "FLMGET" USING FLAMID, RETCO,
                         RECLEN, REC-ORD, BUFLEN
```

```
        IF GAP
           DISPLAY "*** FOUND EMPTY SLOT ***"      UPON TERMOUT
           MOVE    0 TO RETCO
        ELSE
           IF OK OR CUT
              DISPLAY RECORD-DISPLAY               UPON TERMOUT
           END-IF
        END-IF
     END-PERFORM.
     IF NOT OK
        DISPLAY "ERROR ON 'GET RECORD'"            UPON TERMOUT
        PERFORM FEHLER-MELDUNG
     END-IF.
*
 SEQUENTIELL-LESEN-RUECKWAERTS.
*
     DISPLAY "NUMBER RECORDS TO READ ?"            UPON TERMOUT.
     PERFORM NUMERISCHE-EINGABE.
     MOVE    EINGABE-NUM TO GET-COUNT.
     MOVE    0 TO RETCO.
     PERFORM VARYING GET-INDEX FROM 0 BY 1
             UNTIL GET-INDEX = GET-COUNT OR NOT OK
        MOVE    SPACES TO RECORD-DISPLAY
        CALL   "FLMGTR" USING FLAMID, RETCO,
                             RECLEN, REC-ORD, BUFLEN
        IF GAP
           DISPLAY "*** FOUND EMPTY SLOT ***"      UPON TERMOUT
           MOVE    0 TO RETCO
        ELSE
           IF OK OR CUT
              DISPLAY RECORD-DISPLAY               UPON TERMOUT
           END-IF
        END-IF
     END-PERFORM.
     IF NOT OK
        DISPLAY "ERROR ON 'READ BACKWARD'"         UPON TERMOUT
        PERFORM FEHLER-MELDUNG
     END-IF.
*
 SATZNUMMER-LESEN.
*
     DISPLAY " "                                   UPON TERMOUT.
     DISPLAY "RECORD NUMBER ?"                     UPON TERMOUT.
     PERFORM NUMERISCHE-EINGABE.
     MOVE    EINGABE-NUM TO RECNO.
     MOVE    SPACES TO RECORD-DISPLAY
     CALL "FLMGRN" USING FLAMID, RETCO, RECLEN, REC-ORD
                                   BUFLEN, RECNO.
     IF GAP
        DISPLAY "*** FOUND EMPTY SLOT ***"         UPON TERMOUT
        MOVE    0 TO RETCO
     ELSE
        IF OK OR CUT
           DISPLAY RECORD-DISPLAY                  UPON TERMOUT
        END-IF
     END-IF
     IF NOT OK
```

```
        DISPLAY "ERROR ON 'POSITION TO RECORD NUMBER'"
                                                     UPON TERMOUT
        PERFORM FEHLER-MELDUNG
     END-IF.
*
 SATZNUMMER-POSITIONIEREN.
*
     DISPLAY " "                                     UPON TERMOUT.
     DISPLAY "RECORD NUMBER ?"                       UPON TERMOUT.
     PERFORM NUMERISCHE-EINGABE.
     MOVE    EINGABE-NUM TO RECNO.
     DISPLAY "CHECKMODE (0/1/2) ?"                   UPON TERMOUT.
     PERFORM NUMERISCHE-EINGABE.
     MOVE    EINGABE-NUM TO CHECKMODE.
     CALL "FLMFRN" USING FLAMID, RETCO, RECNO, CHECKMODE.
     IF NOT OK
        DISPLAY "ERROR ON 'POSITION TO RECORD NUMBER'"
                                                     UPON TERMOUT
        PERFORM FEHLER-MELDUNG
     ELSE
        DISPLAY "RECORD NUMBER: ", RECNO             UPON TERMOUT
     END-IF.*
 POSITIONIEREN.
*
     DISPLAY " "                                     UPON TERMOUT.
     DISPLAY "RELATIVE POSITION ?"                   UPON TERMOUT.
     PERFORM NUMERISCHE-EINGABE.
     MOVE    EINGABE-NUM TO REL-POSITION.
     CALL "FLMPOS" USING FLAMID, RETCO, REL-POSITION.
     IF NOT OK
        DISPLAY "ERROR ON 'POSITION'"                UPON TERMOUT
        PERFORM FEHLER-MELDUNG
     END-IF.
*
 LOESCHEN.
*
     CALL   "FLMDEL" USING FLAMID, RETCO,
     IF NOT OK
        DISPLAY "ERROR ON 'DELETE'"                  UPON TERMOUT
        PERFORM FEHLER-MELDUNG
     END-IF.
*
 SCHLUESSEL-LESEN.
*
     DISPLAY "RECORD KEY ?"                          UPON TERMOUT.
     MOVE    SPACES TO REC-ORD.
     ACCEPT  RECORD-KEY-DISPLAY                      FROM TERMIN.
     SET     KEY-INDEX      TO   1.
     SET     REC-INDEX      TO   KEYPOS1-ORIG.
     PERFORM VARYING KEY-IND-DISP FROM 0 BY 1
             UNTIL KEY-IND-DISP = KEYLEN1-ORIG
        MOVE  RECORD-KEY-BYTE(KEY-INDEX)  TO  BYTE(REC-INDEX)
        SET   KEY-INDEX  UP  BY  1
        SET   REC-INDEX  UP  BY  1
     END-PERFORM.
     CALL   "FLMGKY" USING FLAMID, RETCO,
                     RECLEN, REC-ORD, BUFLEN.
```

```
     IF NOT OK
     THEN
        DISPLAY "ERROR ON 'READ WITH KEY'"         UPON TERMOUT
        PERFORM FEHLER-MELDUNG
        MOVE    RECORD-KEY-DISPLAY  TO  RECORD-DISPLAY
        DISPLAY "RECORD BUFFER: "                  UPON TERMOUT
        DISPLAY RECORD-DISPLAY                     UPON TERMOUT
     ELSE
        DISPLAY RECORD-DISPLAY                     UPON TERMOUT
     END-IF.
*
 SCHLUESSEL-POSITIONIEREN.
*
     DISPLAY "RECORD KEY ?"                        UPON TERMOUT.
     MOVE    SPACES TO REC-ORD.
     ACCEPT  RECORD-KEY-DISPLAY                    FROM TERMIN.
     DISPLAY "CHECKMODE (0/1/2) ?"                 UPON TERMOUT.
     PERFORM NUMERISCHE-EINGABE.
     MOVE    EINGABE-NUM TO CHECKMODE.
     SET     KEY-INDEX      TO   1.
     SET     REC-INDEX      TO   KEYPOS1-ORIG.
     PERFORM VARYING KEY-IND-DISP FROM 0 BY 1
             UNTIL KEY-IND-DISP = KEYLEN1-ORIG
        MOVE  RECORD-KEY-BYTE(KEY-INDEX)  TO  BYTE(REC-INDEX)
        SET   KEY-INDEX  UP  BY  1
        SET   REC-INDEX  UP  BY  1
     END-PERFORM.
     CALL   "FLMFKY" USING FLAMID, RETCO,
                     RECLEN, REC-ORD, CHECKMODE.
     IF NOT OK
     THEN
        DISPLAY "ERROR ON 'FIND WITH KEY'"         UPON TERMOUT
        PERFORM FEHLER-MELDUNG
        MOVE    RECORD-KEY-DISPLAY  TO  RECORD-DISPLAY
        DISPLAY "RECORD BUFFER: "                  UPON TERMOUT
        DISPLAY RECORD-DISPLAY                     UPON TERMOUT
     END-IF.
*
 SCHREIBEN.
*
     DISPLAY "RECORD LENGTH ?"                     UPON TERMOUT.
     PERFORM NUMERISCHE-EINGABE.
     MOVE    EINGABE-NUM TO RECLEN.
     DISPLAY "RECORD DATA ?"                       UPON TERMOUT.
     MOVE    SPACES TO RECORD-DISPLAY
     ACCEPT  RECORD-DISPLAY                        FROM TERMIN.
     CALL   "FLMPUT" USING FLAMID, RETCO,
                     RECLEN, REC-ORD.
     IF NOT OK
     THEN
        DISPLAY "ERROR ON 'WRITE A RECORD'"        UPON TERMOUT
        PERFORM FEHLER-MELDUNG
     END-IF.
*
 SCHLUESSEL-SCHREIBEN.
*
     DISPLAY "RECORD LENGTH ?"                     UPON TERMOUT.
```

```
      PERFORM NUMERISCHE-EINGABE.
      MOVE    EINGABE-NUM TO RECLEN.
      DISPLAY "RECORD INCLUSIVE KEY ?"              UPON TERMOUT.
      MOVE    SPACES TO RECORD-DISPLAY
      ACCEPT  RECORD-DISPLAY                        FROM TERMIN.
      CALL   "FLMPKY" USING FLAMID, RETCO,
                      RECLEN, REC-ORD.
      IF NOT OK
      THEN
         DISPLAY "ERROR ON 'WRITE WITH KEY'"        UPON TERMOUT
         PERFORM FEHLER-MELDUNG
      END-IF.
 *
  AENDERN.
 *
      DISPLAY "RECORD LENGTH ?"                     UPON TERMOUT.
      PERFORM NUMERISCHE-EINGABE.     MOVE    EINGABE-NUM TO RECLEN.
      DISPLAY "RECORD INCLUSIVE KEY ?"              UPON TERMOUT.
      MOVE    SPACES TO RECORD-DISPLAY
      ACCEPT  RECORD-DISPLAY                        FROM TERMIN.
      CALL   "FLMUPD" USING FLAMID, RETCO,
                      RECLEN, REC-ORD, BUFLEN.
      IF NOT OK
      THEN
         DISPLAY "ERROR ON 'UPDATE'"                UPON TERMOUT
         PERFORM FEHLER-MELDUNG
      END-IF.
 *
  HEADER-SCHREIBEN.
 *
      DISPLAY "DATA SET NAME ?"                     UPON TERMOUT
      ACCEPT   FILENAME-ORIG                        FROM TERMIN
      DISPLAY "NAMELEN (0 - 54) ?"                  UPON TERMOUT
      PERFORM  NUMERISCHE-EINGABE
      MOVE     EINGABE-NUM TO NAMELEN-ORIG
      DISPLAY "DSORG (0=SEQ 1=INDEX 2=REL ...) ?" UPON TERMOUT
      PERFORM  NUMERISCHE-EINGABE
      MOVE     EINGABE-NUM TO DSORG-ORIG
      DISPLAY "RECFORM (0=VAR 1=FIX 2=UNDEF ...) ?" UPON TERMOUT
      PERFORM  NUMERISCHE-EINGABE
      MOVE     EINGABE-NUM TO RECFORM-ORIG
      DISPLAY "RECSIZE (0 - 32768) ?"               UPON TERMOUT
      PERFORM  NUMERISCHE-EINGABE
      MOVE     EINGABE-NUM TO RECSIZE-ORIG
      DISPLAY "BLKSIZE (0 - 32768) ?"               UPON TERMOUT
      PERFORM  NUMERISCHE-EINGABE
      MOVE     EINGABE-NUM TO BLKSIZE-ORIG
      IF   NOT KEYDESC-DEFINIERT
      THEN
           PERFORM  KEYDESC-EINGABE
           MOVE    "N" TO KEYDESC-INDIKATOR
      END-IF
      DISPLAY "PRCTRL (0=NO 1=MACHINE 2=ASA) ?"     UPON TERMOUT
      PERFORM  NUMERISCHE-EINGABE
      MOVE     EINGABE-NUM TO PRCTRL-ORIG
      MOVE     LOW-VALUES  TO SYSTEM-ORIG
      DISPLAY "LASTPAR (0=YES 1=NO) ?"              UPON TERMOUT
```

```
     PERFORM  NUMERISCHE-EINGABE
     MOVE     EINGABE-NUM TO LASTPAR-PHD
*
     CALL   "FLMPHD" USING FLAMID, RETCO,
                     NAMELEN-ORIG, FILENAME-ORIG,
                     DSORG-ORIG, RECFORM-ORIG,
                     RECSIZE-ORIG, RECDELIM-ORIG,
                     KEYDESC-ORIG, BLKSIZE-ORIG,
                     PRCTRL-ORIG, SYSTEM-ORIG,
                     LASTPAR-PHD.
     IF NOT OK
     THEN
          DISPLAY "ERROR ON 'WRITE HEADER'"        UPON TERMOUT
          PERFORM FEHLER-MELDUNG
     ELSE          IF NOT LAST-PARAMETER-PHD
          THEN
               DISPLAY " "                         UPON TERMOUT
               DISPLAY "WRITE USER HEADER: "       UPON TERMOUT
               PERFORM USER-HEADER-SCHREIBEN
          END-IF
     END-IF.
*
 USER-HEADER-SCHREIBEN.
*
     DISPLAY "HEADERLENGTH ?"                      UPON TERMOUT.
     PERFORM NUMERISCHE-EINGABE.
     MOVE    EINGABE-NUM TO UATTRLEN.
     DISPLAY "YOUR INPUT, PLEASE"                  UPON TERMOUT.
     ACCEPT  USERATTR                              FROM TERMIN.
     CALL   "FLMPUH" USING FLAMID, RETCO,
                     UATTRLEN, USERATTR.
     IF NOT OK
     THEN
        DISPLAY "ERROR ON 'WRITE USER HEADER'"     UPON TERMOUT
        PERFORM FEHLER-MELDUNG
     END-IF.
*
 HEADER-LESEN.
*
     MOVE   54       TO NAMELEN-ORIG.
     MOVE   SPACES   TO FILENAME-ORIG.
     CALL   "FLMGHD" USING FLAMID, RETCO,
                     NAMELEN-ORIG, FILENAME-ORIG,
                     DSORG-ORIG, RECFORM-ORIG,
                     RECSIZE-ORIG, RECDELIM-ORIG,
                     KEYDESC-ORIG, BLKSIZE-ORIG,
                     PRCTRL-ORIG, SYSTEM-ORIG.
     IF NOT OK
     THEN
        DISPLAY "ERROR ON 'READ HEADER'"           UPON TERMOUT
        PERFORM FEHLER-MELDUNG
     ELSE
        DISPLAY "NAMELEN  ", NAMELEN-ORIG          UPON TERMOUT
        DISPLAY "DATA SET ", FILENAME-ORIG         UPON TERMOUT
        DISPLAY "DSORG    ", DSORG-ORIG            UPON TERMOUT
        DISPLAY "RECFORM  ", RECFORM-ORIG          UPON TERMOUT
        DISPLAY "RECSIZE  ", RECSIZE-ORIG          UPON TERMOUT
```

```
         PERFORM  KEYDESC-AUSGABE
         DISPLAY "BLKSIZE  ", BLKSIZE-ORIG          UPON TERMOUT
         DISPLAY "PRCTRL   ", PRCTRL-ORIG           UPON TERMOUT
         DISPLAY "RECSIZE  ", RECSIZE-ORIG          UPON TERMOUT
         MOVE     SYSTEM-ORIG TO BYTE-3-4-HEX
         PERFORM  HEX-TO-CHAR
         DISPLAY "SYSTEM   ", BYTE-3-4-CHAR         UPON TERMOUT
      END-IF.
*
 USER-HEADER-LESEN.
*
      MOVE   80       TO UATTRLEN.
      MOVE   SPACES   TO USERATTR.
      CALL   "FLMGUH" USING FLAMID, RETCO,
                      UATTRLEN, USERATTR.
      IF NOT OK
      THEN
         DISPLAY "ERROR ON 'READ USER HEADER'"      UPON TERMOUT
         PERFORM FEHLER-MELDUNG
      ELSE
         DISPLAY "LENGTH USERHEADER ", UATTRLEN     UPON TERMOUT
         IF UATTRLEN  0
         THEN
            DISPLAY USERATTR                        UPON TERMOUT
         END-IF
      END-IF.
*
 MATRIX-ABSCHLIESSEN.
*
      CALL   "FLMFLU" USING FLAMID, RETCO CPUTIME REC-ORDS
                            BYTES BYTEOFL CMPRECS CMPBYTES
                            CMPBYOFL.
      IF NOT OK
         DISPLAY "ERROR ON 'FLUSH MATRIX'  "        UPON TERMOUT
         PERFORM FEHLER-MELDUNG
      ELSE
         IF STATISTIK
         THEN
            DISPLAY " "                             UPON TERMOUT
            MOVE     REC-ORDS  TO STATIS-DIS
            DISPLAY "ORG.  RECORDS ", STATIS-DIS UPON TERMOUT
            MOVE     BYTECNT   TO STATIS-DIS
            DISPLAY "ORG.  BYTES   ", STATIS-DIS UPON TERMOUT
            MOVE     CMPRECS   TO STATIS-DIS
            DISPLAY "COMP. RECORDS ", STATIS-DIS UPON TERMOUT
            MOVE     CMPBYCNT  TO STATIS-DIS
            DISPLAY "COMP. BYTES   ", STATIS-DIS UPON TERMOUT
         END-IF
      END-IF.
*
*****************************************************************
*    HILFSFUNKTIONEN                                            *
*****************************************************************
*
 FEHLER-MELDUNG.
*
      IF   UNZULAESSIG
```

```
     THEN DISPLAY "ILLEGAL FUNCTION"               UPON TERMOUT
     ELSE
          IF   DVS-ERROR
          THEN
               MOVE     LOW-VALUE TO RETCO-INDICATOR
               MOVE     RETCO TO HEXDATA
               PERFORM  HEX-TO-CHAR
               DISPLAY "VSAM ERROR CODE: ", BYTE-2-4-CHAR
                                                   UPON TERMOUT
          ELSE
               DISPLAY "FLAM ERROR CODE: ", RETCO-FLAM
                                                   UPON TERMOUT
          END-IF
     END-IF.
*
 NUMERISCHE-EINGABE.
*
     ACCEPT  EINGABE                        FROM TERMIN.
     MOVE    0 TO EINGABE-NUM.
     SET     RED-INDEX TO 8.
     PERFORM VARYING EIN-INDEX
                FROM 9 BY -1 UNTIL EIN-INDEX = 0
                                OR RED-INDEX = 0
             IF      BYTE-EIN(EIN-INDEX) NUMERIC
             THEN    MOVE BYTE-EIN(EIN-INDEX)
                       TO BYTE-RED(RED-INDEX)
                     SET  RED-INDEX DOWN BY 1
             END-IF
     END-PERFORM.
     IF      BYTE-EIN(1) = "-"
     THEN    COMPUTE EINGABE-NUM = -1 * EINGABE-NUM
     END-IF.
*
 HEX-TO-CHAR.
*
      PERFORM VARYING CHAR-INDEX
                 FROM 8 BY -1 UNTIL CHAR-INDEX = 0
              DIVIDE HEXDATA BY 16 GIVING HEX-QUOTIENT
                 REMAINDER HEX-REMAINDER
              END-DIVIDE
              ADD  1            TO HEX-REMAINDER
              SET  HEX-INDEX    TO HEX-REMAINDER
              MOVE HEX-QUOTIENT TO HEXDATA
              MOVE DIGIT-HEX(HEX-INDEX)
                TO BYTE-CHAR(CHAR-INDEX)
      END-PERFORM.
*
 KEYDESC-EINGABE.
*
      DISPLAY "KEYPARTS (0 - 8) ?"                 UPON TERMOUT
      PERFORM  NUMERISCHE-EINGABE
      MOVE     EINGABE-NUM TO KEYPARTS-ORIG
      IF   KEYPARTS-ORIG  0
      THEN
           DISPLAY "KEYFLAGS (0=NODUP 1=DUPKY) ?"  UPON TERMOUT
           PERFORM  NUMERISCHE-EINGABE
           MOVE     EINGABE-NUM TO KEYFLAGS-ORIG
```

```
           DISPLAY "KEYPOS1 (1 - 32767) ?"         UPON TERMOUT
           PERFORM  NUMERISCHE-EINGABE
           MOVE     EINGABE-NUM TO KEYPOS1-ORIG
           DISPLAY "KEYLEN1 (1 - 255) ?"           UPON TERMOUT
           PERFORM  NUMERISCHE-EINGABE
           MOVE     EINGABE-NUM TO KEYLEN1-ORIG
           DISPLAY "KEYTYPE1 (0=DISP 1=BINARY) ?"  UPON TERMOUT
           PERFORM  NUMERISCHE-EINGABE
           MOVE     EINGABE-NUM TO KEYTYPE1-ORIG
           PERFORM  VARYING KEYDESC-INDEX FROM 1 BY 1
                       UNTIL KEYDESC-INDEX = KEYPARTS-ORIG
                    SET      DIGIT TO KEYDESC-INDEX
                    ADD      1 TO DIGIT
                    DISPLAY "KEYPOS", DIGIT, " (1 - 32767) ?"
                                                   UPON TERMOUT
                    PERFORM  NUMERISCHE-EINGABE
                    MOVE     EINGABE-NUM
                      TO     KEYPOS-ORIG(KEYDESC-INDEX)
                    DISPLAY "KEYLEN", DIGIT, " (1 - 255) ?"
                                                   UPON TERMOUT
                    PERFORM  NUMERISCHE-EINGABE
                    MOVE     EINGABE-NUM
                      TO     KEYLEN-ORIG(KEYDESC-INDEX)
                    DISPLAY "KEYTYPE", DIGIT, " (0=DISP 1=BIN) ?"
                                                   UPON TERMOUT
                    PERFORM  NUMERISCHE-EINGABE
                    MOVE     EINGABE-NUM
                      TO     KEYTYPE-ORIG(KEYDESC-INDEX)
           END-PERFORM
      END-IF.
*
 KEYDESC-AUSGABE.
*
      IF   KEYPARTS-ORIG  0
      THEN
           DISPLAY "KEYDESC OF ORIGINAL DATA SET"  UPON TERMOUT
           DISPLAY "KEYPARTS ", KEYPARTS-ORIG      UPON TERMOUT
           DISPLAY "KEYFLAGS ", KEYFLAGS-ORIG      UPON TERMOUT
           DISPLAY "KEYPOS1  ", KEYPOS1-ORIG       UPON TERMOUT
           DISPLAY "KEYLEN1  ", KEYLEN1-ORIG       UPON TERMOUT
           DISPLAY "KEYTYPE1 ", KEYTYPE1-ORIG      UPON TERMOUT
           PERFORM  VARYING KEYDESC-INDEX FROM 1 BY 1
                       UNTIL KEYDESC-INDEX = KEYPARTS-ORIG
              SET      DIGIT TO KEYDESC-INDEX
              ADD      1 TO DIGIT
              DISPLAY "KEYPOS", DIGIT, "  ",
                       KEYPOS-ORIG(KEYDESC-INDEX)  UPON TERMOUT
              DISPLAY "KEYLEN", DIGIT, "  ",
                       KEYLEN-ORIG(KEYDESC-INDEX)  UPON TERMOUT
              DISPLAY "KEYTYPE", DIGIT, " ",
                       KEYTYPE-ORIG(KEYDESC-INDEX) UPON TERMOUT
           END-PERFORM
      END-IF.
```

5.3 Benutzer Ein-/Ausgabe Schnittstelle

5.3.1 ASSEMBLER-Beispiel

Dieses Beispiel realisiert ein DUMMY-Device, das beim Lesen sofort beim ersten Satz den Returncode END-OF-FILE liefert. Beim Schreiben werden alle S tze bernommen. Es wird immer der Returncode OK zur ckgegeben, ohne dass die S tze irgendwohin geschrieben werden. Die Funktionen USRGKY und USRPOS liefern immer den Returncode INVALID-KEY bzw. INVALID-POSITION. Die Funktion USRDEL liefert immer den Returncode INVALID-FUNCTION.

Diese Funktionalit t entspricht einer Dateizuweisung auf DUMMY.

Durch Ausf llen der mit drei Punkten markierten Sequenzen, kann diese Routine als Ger st f r eine spezielle Benutzer Ein-/Ausgabe Routine benutzt werden.

```
FLAMUIO  START
         TITLE 'FLAMUIO: USER-I/O-MODULE FOR FLAM V2.5'
***********************************************************************
*  NAME: FLAMUIO                                                      *
*  FUNCTION:                                                          *
*        DUMMY MODULE AS EXAMPLE FOR AN USER-IO-MODULE                *
*  INTERFACES:                                                        *
*        USROPN   OPEN DATA SET                                       *
*        USRCLS   CLOSE DATA SET                                      *
*        USRGET   READ SEQUENTIAL                                     *
*        USRGKY   READ WITH KEY                                       *
*        USRPUT   WRITE SEQUENTIAL                                    *
*        USRPKY   WRITE WITH KEY                                      *
*        USRDEL   DELETE ACTUAL RECORD                                *
*        USRPOS   POSITION IN DATA SET                                *
*  NOTES:                                                             *
*        ALL FUNCTIONS ARE REENTRANT.                                 *
*        WE NEED NO RUN TIME SYSTEM.                                  *
*        INDEPENDANT FROM ANY /370-SYSTEM.                            *
***********************************************************************
*
*  ADDRESSING -/ RESIDENCY MODE
*
FLAMUIO  AMODE ANY
FLAMUIO  RMODE ANY
*
*  RETURN CODES
*

OK       EQU   0             KEIN FEHLER
*        EQU   -1            REQM-FEHLER; UNGUELTIGE KENNUNG BZW.
*                            UNZULAESSIGE FUNCTION
CUT      EQU   1             SATZ VERKUERZT
EOF      EQU   2             DATEIENDE
GAP      EQU   3             LUECKE IN RELATIVER DATEI
```

```
FILL     EQU   4             SATZ AUFGEFUELLT
INVKEY   EQU   5             SCHLUESSEL NICHT VORHANDEN
RCEMPTY  EQU   30            EINGABEDATEI IST LEER
RCNEXIST EQU   31            DATEI IST NICHT VORHANDEN
RCOPENMO EQU   32            UNZULAESSIGER OPEN-MODE
RCFCBTYP EQU   33            UNZULAESSIGES DATEIFORMAT
RCRECFOR EQU   34            UNZULAESSIGES SATZFORMAT
RCRECSIZ EQU   35            UNZULAESSIGE SATZLAENGE
RCBLKSIZ EQU   36            UNZULAESSIGE BLOCKGROESSE
RCKEYPOS EQU   37            UNZULAESSIGE SCHLUESSELPOSITION
RCKEYLEN EQU   38            UNZULAESSIGE SCHLUESSELLAENGE
RCDSN    EQU   39            UNZULAESSIGER DATEINAME
*        EQU   X'0FXXXXXX'   SONSTIGER FEHLER
*
***********************************
* REGISTER EQUATES                *
***********************************
R0       EQU   0
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R10      EQU   10
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
*
         DC    C'*** MODULE FLAMUIO. '
         DC    C'USER-I/O-MODULE FOR FLAM '
         DC    C'TIME - DATE ASSEMBLED: '
         DC    C'&SYSDATE - &SYSTIME ***'
         TITLE 'USROPN'
USROPN   DS    0D
         ENTRY USROPN
         USING USROPN,R10

**********************************************************************
*  NAME: USROPN                                                      *
*  FUNCTION:                                                         *
*        OPEN DATA SET                                               *
*  PARAMETER:                                                        *
*  1 <-> WORKAREA 256F   WORKAREA, INITIALIZED WITH X'00'.           *
*                        THIS AREA IS CONNECTED TO THIS DATA SET.    *
*                        USABLE AS WORKAREA DURING THE DIFFERENT CALLS*
*                        FOR THE ACTUAL DATA SET.                    *
*  2 <-  RETCO    F      RETURNCODE                                  *
*        = 0             NO ERROR                                    *
*        = 30            INPUT DATA SET IS EMPTY                     *
*        = 31            DATA SET NOT CONNECTED OR DOES NOT EXIST    *
*        = 32            ILLEGAL OPEN MODE                           *
```

```
*       = 33           ILLEGAL DSORG                                 *
*       = 34           ILLEGAL RECORD FORMAT                         *
*       = 35           ILLEGAL RECORD LENGTH                         *
*       = 36           ILLEGAL BLOCK SIZE                            *
*       = 37           ILLEGAL KEY POSITION                          *
*       = 38           ILLEGAL KEY LENGTH                            *
*       = -1           UNSUPPORTED FUNCTION; GETMAIN ERROR           *
*       = X'0FXXXXXX'  SONSTIGER FEHLERCODE                          *
*  3 ->  OPENMODE F     OPEN MODE                                    *
*       = 0            INPUT (SEQUENTIAL READ)                       *
*                      (DATA SET MUST EXIST)                         *
*       = 1            OUTPUT (SEQUENTIAL WRITE)                     *
*                      (DATA SET WILL BE OVERWRITTEN)                *
*       = 2            INOUT (READ OR WRITE SEQUENTIAL OR WITH KEY) *
*                      (DATA SET MUST EXIST)                         *
*       = 3            OUTIN (WRITE OR READ SEQUENTIAL OR WITH KEY) *
*                      (DATA SET WILL BE OVERWRITTEN)                *
*  4 ->  DDNAME   CL8   DD-NAME                                      *
*  5 <-> DSORG    F     DATA SET ORGANIZATION                        *
*       = 0;  8; 16 ... SEQUENTIAL                                   *
*       = 1;  9; 17 ... INDEX SEQUENTAIL                             *
*       = 2; 10; 18 ... RELATIVE                                     *
*       = 3; 11; 19 ... DIRECT                                       *
*       = 4; 12; 20 ... UNSTRUCTERED                                 *
*       = 5; 13; 21 ... LIBRARY                                      *
*  6 <-> RECFORM  F     RECORD FORMAT                                *
*       = 0;  8; 16 ... VARIABELE (V)                                *
*                      8 = BLOCKED  16 = BLOCKED/SPANNED             *
*       = 1;  9; 17 ... FIX (F)                                      *
*                      9 = BLOCKED  17 = BLOCKED/SPANNED             *
*       = 2; 10; 18 ... UNDEFINED (U)                                *
*                                                                    *
*       = 3; 11; 19 ... STREAM (S)                                   *
*                      11 = DELIMITER    19 RECORD DESCRIPTOR WORD  *
*  7 <-> RECSIZE  F     DATA LENGTH (WITHOUT DELIMTER OR RDW)        *
*       = 0 BIS 32767                                                *
*         RECFORM = V:  MAX. RECORD LENGTH OR 0                      *
*         RECFORM = F:  RECORD LENGTH                                *
*         RECFORM = U:  MAX. RECORD LENGTH OR 0                      *
*         RECFORM = S:  LENGTH DELIMITER OR RDW                      *

*  8 <-> BLKSIZE  F        BLOCK SIZE                                *
*       = 0            UNBLOCKED                                     *
*  9 <-> KEYDESC  STRUCT   KEY DESCRIPTION                           *
*                                                                    *
*       KEYFLAGS F     OPTIONS                                       *
*       = 0            NO DUPLICATE KEYS                             *
*       = 1            DUPLICATES ALLOWED                            *
*       KEYPARTS F     NUMBER OF KEY PARTS                           *
*       = 0 BIS 8                                                    *
*       KEYPOS1  F     1. BYTE OF 1. KEYPART                         *
*       = 1 BIS 32766                                                *
*       KEYLEN1  F     LENGTH OF 1. KEYPART                          *
*       = 1 BIS 255                                                  *
*       KEYTYPE1 F     DATA TYPE OF 1. KEYPART                       *
*       = 0            PRINTABLE CHARACTER                           *
*       = 1            BINARY                                        *
```

```
*        .                                                           *
*        .                                                           *
*        .                                                           *
*        KEYPOS8  F      1. BYTE OF 8. KEYPART                       *
*        = 1 BIS 32766                                               *
*        KEYLEN8  F      LENGTH OF 8. KEYPART                        *
*        = 1 BIS 255                                                 *
*        KEYTYPE8 F      DATA TYPE OF 8. KEYPART                     *
*        = 0             PRINTABLE CHARACTER                         *
*        = 1             BINARY                                      *
*                                                                    *
* 10 <-> DEVICE   F      DEVICE TYPE                                 *
*        = 7; 15; 23     USER DEFINED                                *
* 11 <-> RECDELIM XL     RECORD DELIMITER                            *
* 12 ->  PADCHAR  XL1    PADDIND CHARACTER                           *
* 13 <-> PRCTRL   F      PRINTER CONTROL CHARACTER                   *
*        = 0             NONE                                        *
*        = 1             ASA-CHARACTER                               *
*        = 2             MACHINE SPECIFIC CHARACTER                  *
* 14 ->  CLOSDISP F      CLOSE PROCESSING                            *
*        = 0             REWIND                                      *
*        = 1             UNLOAD                                      *
*        = 2             RETAIN / LEAVE                              *
* 15 ->  ACCESS   F      ACCESS METHOD                               *
*        = 0             LOGICAL (RECORD BY RECORD)                  *
*        = 1             PHYSICAL                                    *
* 16 <-> DSNLEN   F      LENGTH OF DATA SET NAME OR BUFFER FOR NAME  *
* 17 <-> DSN      CL     DATA SET NAME                               *
*                        (DAT SET NAME SHOULD BE RETURNED, IF 1. BYTE *
*                        OF GIVEN NAME IS C' ' OR A DIFFERENT DATA SET*
*                        IS ALLOCATED).                              *
**********************************************************************
*
*  SAVE REGISTER AND LOAD PROGRAM REGISTER
*
         STM   R14,R12,12(R13)
         LR    R10,R15

*
*  LOAD PARAMETER
*
         LM    R1,R2,0(R1)
*
*  ADDRESS WORKAREA
*
         LR    R12,R1
         USING WORKAREA,R12
*
*  OPEN DATA SET
*
*        .
*        .
*        .
*
*  SET RETURNCODE TO 'NO ERROR'
*
         LA    R0,OK
```

```
         ST    R0,0(R2)
*
*  RETURN
*
         LM    R14,R12,12(R13)
         BR    R14
*
*  RELEASE WORKAREAS REGISTER*
         DROP  R12
*
***********************************************************************
*  LOCAL CONSTANTS                                                    *
***********************************************************************
*
         LTORG
         DROP  R10
         TITLE 'USRCLS'
USRCLS   DS    0D
         ENTRY USRCLS
         USING USRCLS,R10
***********************************************************************
*  NAME: USRCLS                                                       *
*  FUNCTION:                                                          *
*        CLOSE DATA SET                                               *
*  PARAMETER:                                                         *
*  1 <-> WORKAREA 256F    WORKAREA                                    *
*  2 <-  RETCO    F        RETURNCODE                                 *
*        = 0              NO ERROR                                    *
*        = -1             UNSUPPORTED FUNCTION                        *
*        = X'0FXXXXXX'    ELSE                                        *
*        OR               DMS-ERROR CODE                              *
***********************************************************************
*
*  SAVE REGISTER AND LOAD PROGRAM REGISTER
*
         STM   R14,R12,12(R13)
         LR    R10,R15
*
*  LOAD PARAMETER
*
         LM    R1,R2,0(R1)
*
*  ADDRESS WORKAREA
*
         LR    R12,R1
         USING WORKAREA,R12
*
*  CLOSE DATA SETR
*
*        .
*        .
*        .
*
*  SET RETURNCODE TO 'NO ERROR'
*
         LA    R0,OK
         ST    R0,0(R2)
```

```
*
*  RETURN
*
         LM    R14,R12,12(R13)
         BR    R14
**  RELEASE WORKAREAS REGISTER
*
         DROP  R12
*
**********************************************************************
*  LOCAL CONSTANTS                                                   *
**********************************************************************
*
         LTORG
         DROP  R10
         TITLE 'USRGET'
USRGET   DS    0D
         ENTRY USRGET
         USING USRGET,R10
**********************************************************************
*  NAME: USRGET                                                      *
*  FUNCTION:                                                         *
*        READ A RECORD (SEQUENTIAL)                                  *
*  PARAMETER:                                                        *
*  1 <-> WORKAREA 256F   WORKAREA                                    *
*  2 <-  RETCO    F      RETURNCODE                                  *
*        = 0             NO ERROR                                    *
*        = 1             RECORD TRUNCATED                            *
*        = 2             END OF FILE                                 *
*        = 3             EMPTY SLOT IN RELATIVE RECORD DATA SET      *
*        = -1            UNSUPPORTED FUNCTION                        *
*        = X'0FXXXXXX'   ELSE                                        *
*  3 <-  RECLEN   F      RECORD LENGTH IN BYTES                      *
*  4 <-  RECORD   XL     RECORD                                      *
*  5 ->  BUFLEN   F      LENGTH OF RECORD BUFFER IN BYTES            *
**********************************************************************
*
*  SAVE REGISTER AND LOAD PROGRAM REGISTER
*
         STM   R14,R12,12(R13)
         LR    R10,R15
*
*  LOAD PARAMETER
*
         LM    R1,R5,0(R1)
*
*  ADDRESS WORKAREA
*
         LR    R12,R1
         USING WORKAREA,R12
*
*  READ A RECORD
*
*        .
*        .
*        .
*
```

```
*   HERE:  RETURNCODE 'END OF FILE'
*
         LA    R0,EOF
         ST    R0,0(R2)*
*   RETURN
*
         LM    R14,R12,12(R13)
         BR    R14
*
*   RELEASE WORKAREAS REGISTER
*
         DROP  R12
*
***********************************************************************
*   LOCAL CONSTANTS                                                   *
***********************************************************************
*
         LTORG
         DROP  R10
         TITLE 'USRGKY'
USRGKY   DS    0D
         ENTRY USRGKY
         USING USRGKY,R10
***********************************************************************
*   NAME: USRGKY                                                      *
*   FUNCTION:                                                         *
*        READ RECORD WITH GIVEN RECORD-KEY                            *
*   PARAMETER:                                                        *
*   1 <-> WORKAREA 256F   WORKAREA                                    *
*   2 <-  RETCO    F      RETURNCODE                                  *
*        = 0              NO ERROR                                    *
*        = 1              RECORD TRUNCATED                            *
*        = 2              END OF FILE                                 *
*        = 5              KEY NOT FOUND                               *
*        = -1             UNSUPPORTED FUNCTION                        *
*        = X'0FXXXXXX'    ELSE                                        *
*   3 <-  RECLEN   F      RECORD LENGTH IN BYTES                      *
*   4 <-  RECORD   XL     RECORD WITH SEARCH KEY                      *
*   5 ->  BUFLEN   F      LENGTH OF RECORD BUFFER IN BYTES            *
***********************************************************************
*
*   SAVE REGISTER AND LOAD PROGRAM REGISTER
*
         STM   R14,R12,12(R13)
         LR    R10,R15
*
*   LOAD PARAMETER
*
         LM    R1,R5,0(R1)
*
*   ADDRESS WORKAREA
*
         LR    R12,R1
         USING WORKAREA,R12
*
*   READ RECORD
*
```



```
*         .*         .
*         .
*
*   HERE: RETURNCODE 'RECORD NOT FOUND'
*
          LA    R0,INVKEY
          ST    R0,0(R2)
*
*   RETURN
*
          LM    R14,R12,12(R13)
          BR    R14
*
*   RELEASE WORKAREAS REGISTER
*
          DROP  R12
*

**********************************************************************
*   LOCAL CONSTANTS                                                  *
**********************************************************************
*
          LTORG
          DROP  R10
          TITLE 'USRPUT'
USRPUT    DS    0D
          ENTRY USRPUT
          USING USRPUT,R10
**********************************************************************
*   NAME: USRPUT                                                     *
*   FUNCTION:                                                        *
*         WRITE A RECORD (SEQUENTIAL)                                *
*   PARAMETER:                                                       *
*   1 <-> WORKAREA 256F   WORKAREA                                   *
*   2 <-  RETCO    F      RETURNCODE                                 *
*         = 0             NO ERROR                                   *
*         = 1             RECORD TRUNCATED                           *
*         = 4             RECORD FILLED WITH PADDING CHARACTER       *
*         = -1            UNSUPPORTED FUNCTION                       *
*         = X'0FXXXXXX'   ELSE                                       *
*   3 ->  RECLEN   F      RECORD LENGTH IN BYTES                     *
*   4 ->  RECORD   XL     RECORD                                     *
**********************************************************************
*
*   SAVE REGISTER AND LOAD PROGRAM REGISTER
*
          STM   R14,R12,12(R13)
          LR    R10,R15
*
*   LOAD PARAMETER
*
          LM    R1,R4,0(R1)
*
*   ADDRESS WORKAREA
*
          LR    R12,R1         USING WORKAREA,R12
*
```

```
*  WRITE THE RECORD
*
*        .
*        .
*        .
*
*  RETURNCODE: 'NO ERROR'
*
         LA    R0,OK
         ST    R0,0(R2)
*
*  RETURN
*
         LM    R14,R12,12(R13)
         BR    R14
*
*  RELEASE WORKAREAS REGISTER
*
         DROP  R12
*
***********************************************************************
*  LOCAL CONSTANTS                                                    *
***********************************************************************
*
         LTORG
         DROP  R10
         TITLE 'USRPKY'
USRPKY   DS    0D
         ENTRY USRPKY
         USING USRPKY,R10
***********************************************************************
*  NAME: USRPKY                                                       *
*  FUNCTION:                                                          *
*        WRITE A RECORD WITH GIVEN KEY  (INDEX SEQUENTIAL)            *
*  PARAMETER:                                                         *
*  1 <-> WORKAREA 256F   WORKAREA                                     *
*  2 <-  RETCO    F      RETURNCODE                                   *
*        = 0             NO ERROR                                     *
*        = 1             RECORD TRUNCATED                             *
*        = 4             RECORD FILLED WITH PADDING CHARACTER         *
*        = 5             INVALID KEY                                  *
*        = -1            UNSUPPORTED FUNCTION                         *
*        = X'0FXXXXXX'   ELSE                                         *
*  3 ->  RECLEN   F      RECORD LENGTH IN BYTES                       *
*  4 ->  RECORD   XL     RECORD                                       *
*  NOTES:                                                             *
*        IF THE GIVEN KEY IS THE SAME THAN THE LAST READ KEY, THE     *
*        RECORD SHALL BE OVERWRITTEN (REWRITE).                       *
*        OTHERWISE, THE RECORD SHALL BE INSERTED.                     *
***********************************************************************
*
*  SAVE REGISTER AND LOAD PROGRAM REGISTER
*
         STM   R14,R12,12(R13)          LR    R10,R15
*
*  LOAD PARAMETER
*
```

```
         LM    R1,R5,0(R1)
*
*  ADDRESS WORK AREA
*
         LR    R12,R1
         USING WORKAREA,R12
*
*  WRITE THE RECORD
*
*        .
*        .
*        .

*
*  RETURNCODE:  'NO ERROR'
*
         LA    R0,OK
         ST    R0,0(R2)
*
*  RETURN
*
         LM    R14,R12,12(R13)
         BR    R14
*
*  RELEASE WORKAREAS REGISTER
*
         DROP  R12
*
***********************************************************************
*  LOCAL CONSTANTS                                                    *
***********************************************************************
*
         LTORG
         DROP  R10
         TITLE 'USRDEL'
USRDEL   DS    0D
         ENTRY USRDEL
         USING USRDEL,R10
***********************************************************************
*  NAME: USRDEL                                                       *
*  FUNCTION:                                                          *
*        DELETE ACTUAL RECORD                                         *
*  PARAMETER:                                                         *
*  1 <-> WORKAREA 256F   WORKAREA                                     *
*  2 <-  RETCO    F      RETURNCODE                                   *
*        = 0             NO ERROR                                     *
*        = 5             NO ACTUAL RECORD READ                        *
*        = -1            UNSUPPORTED FUNCTION                         *
*        = X'0FXXXXXX'   ELSE                                         *
***********************************************************************
*
*  SAVE REGISTER AND LOAD PROGRAM REGISTER*
         STM   R14,R12,12(R13)
         LR    R10,R15
*
*  LOAD PARAMETER
*
```

```
         LM    R1,R2,0(R1)
*
*  ADDRESS WORKAREA
*
         LR    R12,R1
         USING WORKAREA,R12
*
*  DELETE RECORD
*
*        .
*        .
*        .
*
*  HERE:  RETURNCODE 'NO ACTUAL RECORD READ'
*
         LA    R0,INVKEY
         ST    R0,0(R2)
*
*  RUECKSPRUNG
*
         LM    R14,R12,12(R13)
         BR    R14
*
*  RELEASE WORKAREAS REGISTER
*
         DROP  R12
*
***********************************************************************
*  LOCAL CONSTANTS                                                    *
***********************************************************************
*
         LTORG
         DROP  R10
         TITLE 'USRPOS'
USRPOS   DS    0D
         ENTRY USRPOS
         USING USRPOS,R10
***********************************************************************
*  NAME: USRPOS                                                       *
*  FUNCTION:                                                          *
*        POSITION IN DATA SET                                         *
*  PARAMETER:                                                         *
*  1 <-> WORKAREA F       WORKAREA                                    *
*  2 <-  RETCO    F       RETURNCODE                                  *
*        = 0              OK                                          *
*        = 5              ILLEGAL POSITION                            *
*        = -1             UNSUPPORTED FUNCTION                        *
*        = X'0FXXXXXX'    ELSE                                        *
*  3 ->  POSITION F       RELATIVE POSITION                           *
*        = 0              NO NEW POSITION                             *
*        = - MAXINT       TO BEGINNING OF DATA SET                    *
*                         ( -2147483648  OR  X'80000000')             *
*        = + MAXINT       TO END OF DATA SET                          *
*                         ( +2147483647  OR  X'7FFFFFFF')             *
*        = - N            N RECORDS BACKWARD                          *
*        = + N            N RECORD FORWARD                            *
*  NOTES:                                                             *
```

```
*        YOU CAN CREATE EMPTY SLOTS ON FORWARD POSITIONING IN A       *
*        RELATIVE DATA SET ON OUTPUT MODE.                            *
***********************************************************************
*
*  SAVE REGISTER AND LOAD PROGRAM REGISTER
*
         STM   R14,R12,12(R13)
         LR    R10,R15

*
*  LOAD PARAMETER
*
         LM    R1,R5,0(R1)
*
*  ADDRESS WORKAREA
*
         LR    R12,R1
         USING WORKAREA,R12
*
*  POSITION RECORD
*
*        .
*        .
*        .
*
*  HERE:  RETURNCODE -1  UNSUPPORTED FUNCTION
*
         LA    R0,0
         BCTR  R0,0
         ST    R0,0(R2)
*
*  RETURN
*
         LM    R14,R15,12(R13)
         BR    R14
*
*  RELEASE WORKAREAS REGISTER
*
         DROP  R12
*
***********************************************************************
*  LOCAL CONSTANTS                                                    *
***********************************************************************
*
         LTORG
         DROP  R10
         TITLE 'FLAMUIO: DUMMY SECTIONS'
***********************************************************************
*  DUMMY SECTIONS                                                     *
***********************************************************************
*
*
WORKAREA DSECT
***********************************************************************
*  WORKAREA ON DOUBLE WORD BOUNDARY                                   *
***********************************************************************
*
```

```
         DS    XL1024
*
LWORK    EQU   *-WORKAREA          LENGTH; MAXIMAL 1024 BYTES
         EJECT

***********************************************************************
*  DUMMY SECTION                                                      *
***********************************************************************
*
*
OPNPAR   DSECT
***********************************************************************
*  PARAMETERLIST FOR USROPN
*
*  NOTE:    ADDRESSES ARE GIVEN, NOT THE VALUES.
***********************************************************************
ADWORKA  DS    A                   WORKAREA
ADRETCO  DS    A                   RETCO
ADOPMO   DS    A                   OPENMODE
ADDDN    DS    A                   DDNAME
ADDSORG  DS    A                   DSORG
ADRECFO  DS    A                   RECFORM
ADRECSI  DS    A                   RECSIZE
ADBLKSI  DS    A                   BLKSIZE
ADKEYDE  DS    A                   KEYDESC
ADEVICE  DS    A                   DEVICE
ADRECDE  DS    A                   RECDELIM
ADPADC   DS    A                   PADCHAR
ADPRCTL  DS    A                   PRCNTRL
ADCLOSDI DS    A                   CLOSDISP
ADACC    DS    A                   ACCESS
ADDSNLEN DS    A                   LENGTH DSN
ADDSN    DS    A                   DATA SET NAME
         EJECT

***********************************************************************
*  DUMMY SECTION                                                      *
***********************************************************************
*
KEYDESC  DSECT
*
*  KEY DESCRIPTION
*
KEYFLAGS DS    F                   KEYFLAGS
KEYPARTS DS    F                   NUMBER OF KEYPARTS
KEYPOS1  DS    F                   KEYPOSITION OF 1. KEYPART
KEYLEN1  DS    F                   LENGTH      OF 1. KEYPART
KEYTYPE1 DS    F                   DATATYPE    OF 1. KEYPART
KEYPOS2  DS    F
KEYLEN2  DS    F
KEYTYPE2 DS    F
KEYPOS3  DS    F
KEYLEN3  DS    F
KEYTYPE3 DS    F
KEYPOS4  DS    F
KEYLEN4  DS    F
KEYTYPE4 DS    F
```

```
KEYPOS5  DS    F
KEYLEN5  DS    F
KEYTYPE5 DS    F
KEYPOS6  DS    F
KEYLEN6  DS    F
KEYTYPE6 DS    F
KEYPOS7  DS    F
KEYLEN7  DS    F
KEYTYPE7 DS    F
KEYPOS8  DS    F                 KEYPOSITION OF 8. KEYPART
KEYLEN8  DS    F                 LENGTH      OF 8. KEYPART
KEYTYPE8 DS    F                 DATATYPE    OF 8. KEYPART
         END
```

5.3.2 COBOL-Beispiel

Die Benutzer Ein-/Ausgabe kann auch in COBOL oder in einer anderen h heren Programmiersprache geschrieben werden. Das folgende Beispiel realisiert zwei verschiedene Funktionen, die ber den symbolischen Dateinamen (LINKNAME bzw. DDNAME) ausgew hlt werden.

Beim DD-Namen "DATBASE" k nnen 10 S tze mit dem Inhalt: "THIS IS A DATABASE RECORD FROM THE USER-IO" gelesen werden, bevor der Returncode END-OF-FILE gemeldet wird.

Beim DD-Namen "USER..." k nnen 20 S tze mit dem Inhalt: "THIS IS A USER RECORD FROM THE USER-IO" gelesen werden, bevor der Returncode END-OF-FILE gemeldet wird.

Zus tzlich werden in beiden F llen die Aufrufe in der Terminalausgabe protokolliert, so dass die Reihenfolge und Aufrufzeitpunkte der einzelnen Funktionen im Ablaufprotokoll von FLAM sehr gut erkennbar sind.

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.  USERIO.
 AUTHOR.      LIMES DATENTECHNIK GMBH.
*
*  USERIO IS AN EXAMPLE FOR AN USER-I/O-MODULE TO CONNECT
*  TO FLAM.
*
*  THE PROGRAM IS WRITTEN TO SUPPORT 2 DIFFERENT DATA SETS IN
*  THE SAME MODULE, DISTINGUISHED BY THE DD-NAME (DATBASE  OR
*                                                 USER....)
*       ENVIRONMENT DIVISION.
*
 CONFIGURATION SECTION.
*
 SPECIAL-NAMES.
     SYSOUT IS  OUT-PUT.
*
 DATA DIVISION.
*
 WORKING-STORAGE SECTION.
*
 77  ALL-OK               PIC S9(8) COMP VALUE 0.
 77  FUNCTION-ERR         PIC S9(8) COMP VALUE -1.
 77  REC-TRUNCATED        PIC S9(8) COMP VALUE 1.
 77  END-OF-FILE          PIC S9(8) COMP VALUE 2.
 77  REC-NOT-FOUND        PIC S9(8) COMP VALUE 5.
 77  NEW-HEADER           PIC S9(8) COMP VALUE 6.
 77  FILE-EMPTY           PIC S9(8) COMP VALUE 30.
 77  FILE-NOT-EXIST       PIC S9(8) COMP VALUE 31.
 77  OPEN-MODE-ERR        PIC S9(8) COMP VALUE 32.
 77  FILE-NAME-ERR        PIC S9(8) COMP VALUE 39.
```

```
*
 77  EXAMPLE-USER-RECORD  PIC X(72) VALUE
        "THIS IS A USER RECORD FROM THE USER-IO".
 77  EXAMPLE-DATBAS-RECORD PIC X(72) VALUE
        "THIS IS A DATA-BASE RECORD FROM THE USER-IO".
 77  RECLEN               PIC S9(8) COMP VALUE 80.
****************************************************
/
 LINKAGE SECTION.
*
 01  USER-WORK.
     03 W-DDNAME      PIC  X(8).
     03 W-COUNTER     PIC  S9(7) COMP-3.
     03 W-ELSE        PIC  X(1012).
 01  RETCO            PIC S9(8)  COMP.
 01  OPENMODE         PIC S9(8)  COMP.
     88  OP-INPUT     VALUE 0.
     88  OP-OUTPUT    VALUE 1.
 01  DDNAME.
     03 DDNAME-1      PIC  X(4).
     03 FILLER        PIC  X(4).
*
*  IN THIS EXAMPLE WE DO NOT NEED THE FOLLOWING PARAMETER
*
*01  DSORG            PIC S9(8)  COMP.
*01  RECFORM          PIC S9(8)  COMP.
*01  RECSIZE          PIC S9(8)  COMP.
*01  BLKSIZE          PIC S9(8)  COMP.
*01  KEYDESC.
*    03 KEYFLAGS PIC S9(8) COMP.
*    03 KEYPARTS PIC S9(8) COMP.
*    03 KEYENTRY                  OCCURS 8 TIMES.
*       05 KEYPOS  PIC S9(8) COMP.
*       05 KEYLEN  PIC S9(8) COMP.
*       05 KEYTYPE PIC S9(8) COMP.
*01  DEVICE           PIC S9(8)  COMP.
*01  RECDELIM         PIC X(4).
*01  PADCHAR          PIC X.
*01  PRCTRL           PIC S9(8)  COMP.
*01  CLOSMODE         PIC S9(8)  COMP.
*01  ACCESS           PIC S9(8)  COMP.
*01  DSNLEN           PIC S9(8)  COMP.
*01  DATA-SET-NAME    PIC X(44).
*
*  USED FOR READING
*
 01  DATALEN          PIC S9(8)  COMP.
 01  DATA-AREA.
     03 DATA-1        PIC X(72).
     03 DATA-2        PIC X(8).
 01  BUFFLEN          PIC S9(8)  COMP.
*
/
 PROCEDURE DIVISION.
*
 USROPN-MAIN SECTION.
*
```

```
*  OPEN ROUTINE
*
 USROPN-MAIN-1.
     ENTRY "USROPN"  USING  USER-WORK, RETCO,
                            OPENMODE, DDNAME.
*
*  IN THIS EXAMPLE WE DO NOT USE THE OTHER PARAMETER, SO IT IS
*  NOT NECESSARY TO MENTION THEM.
*  FLAM STANDARDS ARE USED:
*     SEQUENTIAL,
*     VARIABLE LENGTH UP TO 32752 BYTE (BUT WE ONLY USE 80 BYTE)
*
*
*  WE ONLY SUPPORT OPEN INPUT IN THIS EXAMPLE,
*  CHECK THE OPEN MODE
*
     IF  OP-INPUT
        THEN   NEXT SENTENCE
        ELSE   MOVE  OPEN-MODE-ERR  TO  RETCO
               DISPLAY "USER-IO CANNOT WRITE TO " DDNAME
                       UPON OUT-PUT
               GO TO USROPN-MAIN-99.
*
*  FOR FURTHER USE, WE STORE THE DD-NAME IN THE
*  GIVEN WORKAREA
*
     MOVE  DDNAME    TO   W-DDNAME.
*
*  WE SUPPORT DIFFERENT DATA SETS,
*  CHECK FOR DDNAME "DATBASE", OR THE FIRST 4 BYTE FOR "USER"
*
     IF  DDNAME  =  "DATBASE"              THEN  PERFORM  OPN-DATBASE
        ELSE  IF  DDNAME-1 =  "USER"
               THEN  PERFORM  OPN-USER
               ELSE  MOVE FILE-NAME-ERR  TO  RETCO
                     DISPLAY "USER-IO DOES NOT SUPPORT " DDNAME
                             UPON  OUT-PUT.
 USROPN-MAIN-99.
*
*  GO BACK TO FLAM
*
     GO BACK.
/
 OPN-DATBASE SECTION.
*
*  OPEN-ROUTINE FOR A DATA BASE
*
 OPN-DATBASE-1.
*
*  HERE YOU HAVE TO PROCESS THE OPEN,
*
*
*  INITIALIZE COUNTER-FIELD IN WORK AREA
*
     MOVE  ZERO     TO  W-COUNTER.
*
*  WE ONLY DISPLAY A MESSAGE
```

```
      *
           DISPLAY "USER-IO: OPEN FOR DATABASE IS DONE"
                   UPON  OUT-PUT.
       OPN-DATBASE-90.
      *
      *  SET THE RETURNCODE
      *
           MOVE  ALL-OK   TO  RETCO.
       OPN-DATBASE-99.
           EXIT.
      /
       OPN-USER SECTION.
      *
      *  OPEN-ROUTINE FOR THE OTHER EXAMPLE
      *
       OPN-USER-1.
      *
      *  HERE YOU HAVE TO PROCESS THE OPEN,
      *
      *  INITIALIZE COUNTER-FIELD IN WORK AREA
      *
           MOVE  ZERO     TO  W-COUNTER.
      *
      *  WE ONLY DISPLAY A MESSAGE
      *
           DISPLAY "USER-IO:  OPEN FOR " DDNAME " IS DONE"
                   UPON  OUT-PUT.
       OPN-USER-90.
      *
      *  SET THE RETURNCODE
      *
           MOVE  ALL-OK   TO  RETCO.
       OPN-USER-99.
           EXIT.
      /
       USRCLS-MAIN SECTION.
      *
      *  CLOSE ROUTINE
      *
       USRCLS-MAIN-1.
           ENTRY "USRCLS"  USING  USER-WORK, RETCO.
      *
      *  WE SUPPORT DIFFERENT DATA SETS,
      *  CHECK FOR DDNAME
      *
           IF  W-DDNAME  =  "DATBASE"
              THEN  PERFORM  CLS-DATBASE
              ELSE  PERFORM  CLS-USER.
       USRCLS-MAIN-99.
      *
      *  GO BACK TO FLAM
      *
           GO BACK.
      /
       CLS-USER SECTION.
      *
      *  CLOSE-ROUTINE FOR THE OTHER EXAMPLE
```

```
      *
       CLS-USER-1.
      *
      *  HERE YOU HAVE TO PROCESS THE CLOSE,
      *
      *  WE ONLY DISPLAY A MESSAGE
      *
           DISPLAY "USER-IO:  CLOSE FOR " W-DDNAME " IS DONE"
                   UPON  OUT-PUT.
       CLS-USER-90.
      *
      *  SET THE RETURNCODE
      *
           MOVE  ALL-OK   TO  RETCO.
       CLS-USER-99.
           EXIT.
      /
       CLS-DATBASE SECTION.
      *
      *  CLOSE-ROUTINE FOR A DATA BASE
      *
       CLS-DATBASE-1.
      *
      *  HERE YOU HAVE TO PROCESS THE CLOSE,
      *
      *  WE ONLY DISPLAY A MESSAGE
      *
           DISPLAY "USER-IO: CLOSE FOR DATABASE IS DONE"
                   UPON  OUT-PUT.
       CLS-DATBASE-90.
      *
      *  SET THE RETURNCODE
      *
           MOVE  ALL-OK   TO  RETCO.
       CLS-DATBASE-99.
           EXIT.
      /
       USRGET-MAIN SECTION.
      *
      *  ROUTINE FOR READING RECORDS
      *
       USRGET-MAIN-1.
           ENTRY "USRGET"  USING  USER-WORK, RETCO,
                                  DATALEN, DATA-AREA, BUFFLEN.
      *
      *  WE SUPPORT DIFFERENT DATA SETS,
      *  CHECK FOR DDNAME
      *
           IF  W-DDNAME  =   "DATBASE"
              THEN  PERFORM  GET-DATBASE
              ELSE  PERFORM  GET-USER.
       USRGET-MAIN-99.
      *
      *  GO BACK TO FLAM
      *
           GO BACK.
      /
```

```
 GET-DATBASE SECTION.
*
*  GET-ROUTINE FOR A DATA BASE
*
 GET-DATBASE-1.
*
*  WE RETURN ALWAYS THE SAME RECORD
*
*  AFTER THE 10. RECORD WE FINISH (EOF)
*
     IF  W-COUNTER   < +10
        THEN MOVE  EXAMPLE-DATBAS-RECORD  TO  DATA-1
             MOVE  W-DDNAME               TO  DATA-2
             MOVE  RECLEN                 TO  DATALEN
             ADD   +1                     TO  W-COUNTER
             MOVE  ALL-OK                 TO  RETCO
        ELSE MOVE  ZERO          TO  DATALEN
             MOVE  END-OF-FILE   TO  RETCO.
 GET-DATBASE-99.
     EXIT.
/
 GET-USER SECTION.
*      *  GET-ROUTINE FOR THE OTHER EXAMPLE,
*
 GET-USER-1.
*
*  WE RETURN ALWAYS THE SAME RECORD,
*
*  AFTER THE 20. RECORD WE FINISH (EOF)
*
     IF  W-COUNTER   < +20
        THEN MOVE  EXAMPLE-USER-RECORD  TO  DATA-1
             MOVE  W-DDNAME             TO  DATA-2
             MOVE  RECLEN               TO  DATALEN
             ADD   +1                   TO  W-COUNTER
             MOVE  ALL-OK               TO  RETCO
        ELSE MOVE  ZERO          TO  DATALEN
             MOVE  END-OF-FILE   TO  RETCO.
 GET-USER-99.
     EXIT.
```

5.4 Verwendung der Benutzerausg nge

5.4.1 EXK10/EXD10-Schnittstelle

Die folgende Exitroutine kann sowohl beim Komprimieren als auch beim Dekomprimieren eingesetzt werden. Sie erm glicht das Bearbeiten von Feldern innerhalb von S tzen.

```
         TITLE 'SEPARATE: EXIT ZUR FLAM-KOMPRIMIERUNG'
SEPARATE CSECT
SEPARATE AMODE ANY
SEPARATE RMODE ANY
**********************************************************************
*        DAS PROGRAMM TRENNT FELDER IN DATENSAETZEN, DIE DURCH EIN
*        TRENNZEICHEN SEPARIERBAR SIND, IN EINZELNE FLAM-SAETZE.
*        DADURCH WIRD EINE BESSERE KOMPRIMIERUNG ERREICHT.
*        DAS PROGRAMM IST SO AUSGELEGT, DASS DURCH AENDERUNG IN EINEM
*        STATEMENT EIN ANDERES, AUCH IN DER LAENGE UNTERSCHIEDLICHES
*        TRENNZEICHEN DEFINIERT WERDEN KANN, OHNE DASS DAS PROGRAMM
*        IM ABLAUF GEAENDERT WERDEN MUSS.
*
*        DIE TRENNZEICHEN WERDEN AUS DEM DATENSATZ ELIMINIERT UND DURCH
*        FLAM-SYNTAX ERSETZT.
*        ENTHAELT DER DATENSATZ KEIN TRENNZEICHEN, SO WIRD DER
*        SATZ UNVERAENDERT AN FLAM ZURUECKGEGEBEN.
*
*        SEPARATE WIRD DURCH PARAMETEREINGABE 'EXK10=SEPARATE' BEIM
*        AUFRUF VON FLAM/FLAMUP AKTIVIERT.
*
*        DIE FELDER BESTEHEN AUS ABDRUCKBAREN ZEICHEN, GETRENNT
*        DURCH EIN 2 BYTE LANGES TRENNZEICHEN (X'0D25')
*
*        DIE SO KOMPRIMIERTEN DATEN WERDEN MITTELS FILE TRANSFER ZU
*        EINEM PC UEBERTRAGEN UND MIT FLAM FELDWEISE (MIT TRENNZEICHEN
*        DES JEWEILIGEN BETRIEBSSYSTEMS, WIE X'0D0A' BEI MSDOS ODER
*        NUR X'0A' BEI UNIX) AUF DAS SPEICHERMEDIUM DEKOMPRIMIERT.
*
*  ANMERKUNG:
*
*        BEI DEKOMPRIMIERUNG AUF DEM HOST-RECHNER IST IN EINE
*        DATEI VARIABLER SATZLAENGE ANZUGEBEN.
*        JEDES BEI DER KOMPRIMIERUNG GETRENNTE FELD WIRD IN EINEM
*        SEPARATEN DATENSATZ AUSGEGEBEN. DIE TRENNZEICHEN SIND NICHT
*        MEHR IM SATZ ENTHALTEN.
*        D.H. AUF GROSSRECHNERN IST DIE URSPRUNGSDATEI NICHT
*        REKONSTRUIERBAR.
*
*        DIESER MODUL IST REENTRANT UND REUSABLE
*
*---------------------------------------------------------------------
*
*  AUTOR:          LIMES DATENTECHNIK GMBH
*                  PHILIPP-REIS-PASSAGE 2
*                  D-61381 FRIEDRICHSDORF/TS.
```

```
*                  TEL. 06172-59190
*                  FAX  06172-591939
***********************************************************************
*
*  INTERFACE:  R1 ZEIGT AUF EINE PARAMETERLISTE
*
*   0(R1)   -   A(FUNKTIONSCODE)
*   4(R1)   -   A(RETURNCODE)
*   8(R1)   -   A(A(SATZ))  SATZPOINTER
*  12(R1)   -   A(SATZLAENGE)
*  16(R1)   -   A(WORKAREA)
*
***********************************************************************
         EJECT
         STM   R14,R12,12(R13)    SICHERN REGISTER
         LR    R12,R15            BASISADRESSE IST EINSPRUNGADRESSE
         USING SEPARATE,R12       BASIS REGISTER ZUWEISEN
         USING WORKAREA,R2        BASIS REGISTER WORKAREA
         LA    15,0               ZUNAECHST IST RETURNCODE 0
*
         L     R3,0(,R1)          A(FC LADEN)
         CLC   0(4,R3),FCSATZ     SATZ UEBERGEBEN ?
         BE    SATZUEB            == JA
         CLC   0(4,R3),FCOPEN     OPEN ?
         BNE   RET                == NEIN
*
*  ZUM OPEN ZEITPUNKT WORKAREA-FELDER LOESCHEN
*
         L     R2,16(,R1)        A(WORKAREA)
         MVI   FLAG,X'00'        FLAGS LOESCHEN
         B     RET
SATZUEB  DS    0H

*
*        SATZ WURDE UEBERGEBEN
*
         L     R10,8(,R1)        A(A(SATZ)) NACH R10
         L     R4,0(,R10)        A(SATZ) LADEN
         L     R11,12(,R1)       A(SATZLAENGE)
         L     R5,0(,R11)        SATZLAENGE LADEN
         LA    R9,0(R5,R4)       A(SATZENDE)
         L     R2,16(,R1)        A(WORKAREA)
*
         TM    FLAG,SATZDA       SATZ SCHON GEHABT ?
         BNO   BEGINN            == NEIN
         TM    FLAG,LOESCH       SATZ ZU LOESCHEN ?
         BO    LOESATZ           == JA
*
BEGINNA  DS    0H                SATZ WURDE SCHON BEARBEITET
         L     R4,SATZPTR        A(FELD) VOM LETZTEN MAL
*
BEGINN   DS    0H
         OI    FLAG,SATZDA       KZ FUER SATZ SCHON GEHABT
         LR    R7,R4             A(FELDANFANG SICHERN)
         LR    R6,R9             A(FELDENDE)
         SR    R6,R7             - A(FELDANFANG) = L'RESTSATZ
         BZ    LEERSATZ          L'= 0, LEERSATZ UEBERGEBEN
```

```
         C     R6,LTRENNKZ
         BNL   SUCH              L' < L'TRENNZEICHEN HAT KEIN TRENN-Z.
         OI    FLAG,LOESCH       KZ ZUM LOESCHEN BEI NAECHSTEM RUN
         LR    R4,R9             A(SATZENDE)
         B     SUCHEND
SUCH     DS    0H
         LA    R8,1              SCHRITTWEITE FUER BX-BEFEHL
         S     R9,LTRENNKZ       WG. BX-BEFEHL SATZENDE -L' SETZEN
SUCHLOOP DS    0H
*
*  SUCHKRITERIUM IST (TRENNKZ)
*
         CLC   0(L'TRENNKZ,R4),TRENNKZ   TRENNZEICHEN ?
         BE    ISTDA             == JA
         BXLE  R4,R8,SUCHLOOP    NAECHSTES ZEICHEN
*
         OI    FLAG,LOESCH       KZ ZUM LOESCHEN BEI NAECHSTEM RUN
         LA    R4,L'TRENNKZ-1(R4) FELD IST UM L'-1 GROESSER
         B     SUCHEND
*
ISTDA    DS    0H
         LA    R6,L'TRENNKZ(R4)  SATZPOINTER ERHOEHEN
         ST    R6,SATZPTR        SATZPOINTER SICHERN
SUCHEND  DS    0H

*
*  PARAMETERLEISTE VON FLAM VERSORGEN
*
         SR    R4,R7             FELDLAENGE
         ST    R4,0(R11)         IST SATZLAENGE FUER FLAM
         ST    R7,0(R10)         SATZADRESSE FUER FLAM
         LA    R15,8             RETURNCODE: SATZ EINFUEGEN
*
RET      DS    0H
*
*        ZURUECK ZU FLAM
*
         L     R3,4(,R1)          A(RC) LADEN
         ST    R15,0(,R3)         RC UEBERGEBEN
         L     R14,12(R13)        REGISTER ZURUECKLADEN
         LM    R0,R12,20(R13)
         BR    R14                RUECKSPRUNG
*
LOESATZ  DS    0H
         LA    R15,4              RETURNCODE: SATZ LOESCHEN
         MVI   FLAG,X'00'         FLAG LOESCHEN
         B     RET                UND FERTIG
*
LEERSATZ DS    0H                 NACH TRENNZEICHEN AM SATZENDE
         OI    FLAG,LOESCH        KZ ZUM LOESCHEN BEI NAECHSTEM RUN
         LA    R4,0               SATZ IST LEER
         ST    R4,0(R11)          SATZLAENGE FUER FLAM
         LA    R15,8              RETURNCODE: SATZ EINFUEGEN
         B     RET                UND FERTIG
*
*  KONSTANTEN UND WORKBEREICHE
*
```



```
*
FCSATZ   DC    F'4'                FUNCTION CODE SATZUEBERGABE
FCOPEN   DC    F'0'                              OPEN
LTRENNKZ DC    A(L'TRENNKZ)        LAENGE DES TRENNZEICHENS
*---------------------------------------------------------------------
*
*  BEI ANDEREM TRENNZEICHEN HIER MODIFIZIEREN
*
TRENNKZ  DC    XL2'0D25'           ZU SUCHENDES TRENNZEICHEN
*---------------------------------------------------------------------

*
*  REGISTER
*
R0       EQU   0
R1       EQU   1                   PARAMETER ADRESSE
R2       EQU   2                   BASISREGISTER FUER WORKAREA
R3       EQU   3
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R10      EQU   10
R11      EQU   11
R12      EQU   12                  BASIS REGISTER
R13      EQU   13                  A(SAVE AREA)
R14      EQU   14                  RUECKSPRUNGADRESSE
R15      EQU   15                  EINSPRUNGADRESSE
*
         LTORG
         DC    C'*** MODULE SEPARATE V1.02 FOR FLAM V2.5.'
         DC    C' COPYRIGHT (C) 1990-91 BY LIMES DATENTECHNIK GMBH. '
         DC    C'DATE, TIME ASSEMBLED: '
         DC    C'&SYSDATE , &SYSTIME '
         DC    C'***'
*
*  WORKAREA BEREICH WIRD VON FLAM UEBERGEBEN (1024 BYTE)
*
WORKAREA DSECT
*
DDNAME   DS    CL8                 DD-NAME DER AKTUELLEN DATEI
SATZPTR  DS    A                   SATZPOINTER
FLAG     DS    X                   KENNZEICHEN ZUR VERARBEITUNG
SATZDA   EQU   1                   SATZ WAR SCHON UEBERGEBEN
LOESCH   EQU   2                   SATZ IST ZU LOESCHEN
         END
```

5.4.2 EXK20/EXD20-Schnittstelle

Da FLAM Komprimate mit Checksummen gegen Manipulation sch tzt, lassen sich mit geringstem Aufwand Verschl sselungen in den Benutzerausg ngen f r die Komprimate durchf hren.

Weil das Komprimat bereits verschleiert ist, kann das einfache deterministische Vertauschen von Zeichen im Komprimat von einem unberechtigten Benutzer nur sehr schwer erkannt werden.

Bei der Dekomprimierung f hrt die Vertauschung, sofern sie nicht von einem berechtigten Benutzer r ckg ngig gemacht wird, zu einem Checksummenfehler und das Komprimat kann nicht gelesen werden.

Durch die Symmetrie der Schnittstellen kann bei der Verschl sselung und Entschl sselung die gleiche Routine benutzt werden, sofern die zweimalige Anwendung der gleichen Funktion den Ausgangszustand wiederherstellt, wie das beim Vertauschen der Fall ist.

˜ hnliche Ergebnisse kann man durch bersetzungstabellen erzielen, die mehrere Zeichen paarweise zyklisch vertauschen.

```
         TITLE 'EX20 (B) " VERSION 1.00:06/25/91 " '
* SYMBOLIC CONDITIONS FOR #IF,#WHEN,#WHIL(E),#TOR,#AND,#OR
#LT      EQU   4   LESS THAN
#GT      EQU   2   GREATER THAN
#EQ      EQU   8   EQUAL
#NE      EQU   7   NOT EQUAL
#LE      EQU   13  LESS OR EQUAL
#GE      EQU   11  GREATER OR EQUAL
#LZ      EQU   4   LESS THAN ZERO
#GZ      EQU   2   GREATER THAN ZERO
#ZE      EQU   8   ZERO
#NZ      EQU   7   NOT ZERO
#ON      EQU   1   ONES
#MI      EQU   4   MIXED
#ZO      EQU   11  ZEROS OR ONES
#ZM      EQU   14  ZEROS OR MIXED
#OM      EQU   7   ONES OR MIXED
#F       EQU   15  TRUE IN ANY CASE

* FLOATING POINT REGISTERS, GENERAL REGISTERS, COLUMBUS REGISTERS
FA       EQU   0
FB       EQU   2
FC       EQU   4
FD       EQU   6
R0       EQU   0
R1       EQU   1
R2       EQU   2
R3       EQU   3
R4       EQU   4
R5       EQU   5
```

```
R6       EQU   6
R7       EQU   7
R8       EQU   8
R9       EQU   9
R10      EQU   10
R11      EQU   11
R12      EQU   12
R13      EQU   13
R14      EQU   14
R15      EQU   15
R#PAR    EQU   R1
R#BASE   EQU   R10
R#STACK  EQU   R13
R#EXIT   EQU   R14
R#PASS   EQU   R15
         EJECT
EX20     CSECT
         USING EX20,R#PASS
***********************************************************************
*  NAME: EX20                                      VERSION: 13.03.91 *
*  FUNKTION:                                                          *
*        FLAMFILE AUF EINFACHE WEISE VER- UND ENTSCHLUESSELN          *
*                                                                     *
*        DAS 16.TE UND 17.TE ZEICHEN WIRD VERTAUSCHT. DADURCH         *
*        VERAENDERT SICH DIE CHECKSUMME UND KOMPRIMAT KANN NUR        *
*        VERARBEITET WERDEN, WENN DIE ZEICHEN ZUVOR ERNEUT            *
*        GETAUSCHT WERDEN.                                            *
*  PARAMETER                                                          *
* 1 ->   ID       F    KENNZEICHEN                                    *
* 2 <-   RETCO    F    RETURNCODE                                     *
* 3 ->   RECPTR   A    SATZZEIGER                                     *
* 4 ->   RECLEN   F    SATZLAENGE                                     *
***********************************************************************

*
*  REGISTER SICHERN UND BASISREGISTER LADEN
*
         STM   R14,R12,12(R13)
*
*  PARAMETER LADEN
*
         LM    R1,R4,0(R1)*  KOMPRIMATSSATZ UEBERGEBEN
         CLC   0(4,R1),F4
         BC    #F-#EQ,#F1001
*  SATZLAENGE LADEN
         L     R4,0(R4)
*  SATZLAENGE GROESSER ALS 16
         LA    R14,16
         CR    R4,R14
         BC    #F-#GT,#F1002
*
*  VERTAUSCHEN DES 16.TEN UND 17.TEN ZEICHENS
*
         L     R3,0(R3)
         LA    R14,0(R3,R14)
         IC    R5,0(R14)
         MVC   0(1,R14),1(R14)
```

```
         STC   R5,1(R14)
#F1002   DS    0H
#F1001   DS    0H
*
*  RETURNCODE = SATZ UEBERNEHMEN, BZW OHNE FEHLER
*
         LA    R0,0
         ST    R0,0(R2)
*
*  RUECKSPRUNG
*
         LM    R14,R12,12(R13)
         BR    R#EXIT
*
*  LOKALE KONSTANTEN
*
F4       DC    F'4'
F16      DC    F'16'
         LTORG
         DS    0D
         DROP  R#PASS
         END
```

5.5 Kopplung von FLAM mit anderen Produkten

Mit einigen Herstellern anderer Softwareprodukte wurden gemeinsame Interfaceprogramme zur Kopplung der Software entwickelt.

5.5.1 Kopplung mit NATURALfi

In Zusammenarbeit mit der Software AG wurde f r NATURAL eine Kopplung zu FLAM entwickelt.

NATURAL ist seit der Version 2.2 in der Lage, seine Workfiles und Druckdateien mit FLAM zu schreiben und zu lesen. Damit ist es m glich, mit NATURAL-Programmen komprimierte Dateien zu erzeugen oder zu verarbeiten. Dabei werden auch Dateiformate unterst tzt, die bisher als Workfile nicht zugelassen waren (VSAM-Dateien).

Die Steuerung eines FLAM Einsatzes erfolgt ber JCL (anderer DD-Name), eine ˜ nderung eines NATURAL-Programms ist nicht erforderlich.

Der f r FLAM n tige Modul NATFLAM ist Bestandteil jeder Auslieferung von FLAM f r alle /390-Systeme und muss mit dem zugeh rigen Programm der Software AG zusammengebunden werden.

F r weitere Informationen wenden Sie sich bitte an Ihren Vertriebspartner oder direkt an die Hersteller Software AG und limes datentechnik gmbh.

5.5.2 Kopplung mit SIRONfi

In Zusammenarbeit mit der Ton Beller GmbH in Bensheim wurde f r das Produkt SIRON ein Zugriffsmodul f r FLAM entwickelt.

Damit ist es m glich, mit SIRON-Abfragen komprimierte Dateien mit FLAM zu erzeugen oder zu verarbeiten.

Der ˜ nderungsaufwand bestehender SIRON-Abfragen ist gering, bzw. entf llt durch Eintrag von FLAM im GENAT f r die jeweilige Datei.

JCL-˜ nderungen sind nicht erforderlich.

Entweder wird die NIMM-Schnittstelle verwendet:

```
HOLE datei (NIMM=HZFLAM)
LIES datei (NIMM=HZFLAM),
SCHREIBE datei ... (NIMM=HZFLAM)
```

oder im GENAT-Eintrag f r den DD-Namen der Datei angegeben:

HIN ddname ... MODUL= HZFLAM

Mit dem GENAT-Eintrag werden bei jedem Zugriff auf die Datei die Daten komprimiert oder dekomprimiert.

Der n tige Modul HZFLAM wird durch die Ton Beller GmbH ausgeliefert. Vor Einsatz ist er mit den FLAM-Modulen zusammenzubinden.

F r weitere Informationen wenden Sie sich bitte an Ihren Vertriebspartner oder direkt an die Hersteller Ton Beller GmbH und limes datentechnik gmbh.

FLAM (VSE)

Benutzerhandbuch

Kapitel 6:

Installation

Inhalt

6. Installation

6.1 Installation von FLAM

Vor Beginn der FLAM-Installation muss der Anwender mit LIBR eine FLAM-Library einrichten. Dabei empfiehlt es sich, die Library mit der JCL-Anweisung

// OPTION STDLABEL=ADD

in die Systemstandardlabelarea einzutragen.

In der Regel erfolgt die Auslieferung mittels einer CD.

Die Installation von CD ist mit IND$FILE oder einem kompatiblen Produkt m glich.

Dabei werden die Module mit SEND vom PC auf den VSE-Rechner in die o.a. VSE-LIBR-Bibliothek geladen.

Auf der CD ist zum Laden der Module die Prozedur

SendToHost.cmd

vorbereitet. Bei dieser Prozedur m ssen folgende SET-Parameter angepasst werden.

SET TRM=termid	Terminal-ID der Terminalemulation (meist A).
SET SPATH=sourcepath	Quellpfad der CD (oftmals Z:)
SET LIB=flamlibname	Name der FLAM-Bibliothek
SET SLIB=sublibname	Sublibname der FLAM-Bibliothek
SET PATH=CM-Path	Quellpfad der Terminal Communication Software, in der sich die Programme SEND und RECEIVE befinden.

Nach erfolgreicher bertragung aller Dateien muss FLAM noch lizenziert werden (siehe nachfolgendes Kapitel).

6.2 Lizenzierung von FLAM

FLAM ist gegen unberechtigte Nutzung gesch tzt. Die berechtigte Nutzung von FLAM ist nur mit Hilfe einer von limes datentechnik gmbh vergebenen Lizenznummer m glich. Nur mit dieser Lizenznummer kann FLAM erfolgreich gestartet werden.

Eine Lizenznummer gestattet die Benutzung von FLAM auf einem oder mehreren Rechnern.

Es wird unterschieden zwischen zeitlich befristeten Testlizenzen und zeitlich unbeschr nkten Nutzungslizenzen.

Eine Testlizenz gestattet die Erprobung von FLAM mit allen Funktionen f r einen festgelegten Zeitraum (z.B. 30 Tage).

Die Testprogramme d rfen nicht an Dritte weitergegeben werden.

Mit den Testprogrammen d rfen w hrend der Testzeit keine Daten archiviert werden.

Nach Ablauf der Testzeit sind alle Testprogramme zu l schen.

Eine Nutzungslizenz gestattet die unbefristete Nutzung von FLAM auf den Rechnern, f r die die Lizenz erteilt wurde.

FLAM ist mit einer Sperre versehen, die die unberechtigte Nutzung erkennt und behindert. Das Kopieren von FLAM von einem Rechner auf einen anderen ist nicht gestattet.

Die Schutzmechanismen zur Verhinderung einer Lizenzverletzung sind nach Gesichtspunkten der Praktikabilit t in einer Rechenzentrumsorganisation entwickelt worden. Eine technisch m gliche, vertragswidrige Nutzung ist deshalb noch keine zul ssige Nutzung im Sinne der Lizenzvereinbarung.

FLAM komprimiert strukturorientiert nach dem Algorithmus, der auch Bestandteil des in der Bundesrepublik Deutschland und in den USA sowie beim europ ischen Patentamt patentierten Frankenstein-Limes-Verfahrens ist, angemeldet durch die Erfinder am 19.07.1985.

FLAMfi und FLAMFILEfi sind eingetragene Warenzeichen/ international trademarks.

Verwenden Sie zum Lizenzieren von FLAM den Job

JFLAMLIZ.Z

aus der FLAM-Library oder von CD. Dort m ssen die **Lizenzierungsparameter** und in der SETPARM-Anweisung die flamlib.sublib eingetragen werden.

Die Lizenzierungsparameter erhalten Sie von:

limes datentechnik gmbh
Tel. 06172-5919-0
Fax. 06172-5919-39
info@flam.de

F hren Sie nun den Lizenzierungsjob aus. Bei Returncode=0 ist FLAM einsatzf hig.

```
* $$ JOB JNM=JFLAMLIZ,CLASS=A,DISP=D,PRI=3,NTFY=YES,LDEST=*
* $$ LST DISP=D,CLASS=A,PRI=3
// JOB JFLAMLIZ CREATE LICENSE MODULE FOR FLAM
// STDOPT LINES=66
// OPTION PARTDUMP
// SETPARM FLAMLIB='FLAM.LIB'
* **************************************************************
* ** CREATE LICENSE MODULE *FLAMLIZ.OBJ* IN USER SUBLIBRARY  ** *
* **************************************************************
// LIB          SEARCH=(&FLA
// EXEC FLICENSE,SIZE=AUTO,PARM='&FLAMLIB'
* ****************************************************
* ** L        ARAMETER:                              **
* ** P        TER    LICENSE NUMBER AND LICENSE DATE **
* **                 OR LICENSE NUMBER AND CPU-ID(S)    **
* ****        ****************************************
*    I        UMBER (12)    LICENSE DATE (YYYYDDD) IF TESTLICENSE
*    |        ---|           |-----|
LIC#=
*    CPU-ID (10)   WITHOUT FIRST 2 DIGITS (FF)
*    |--------|
CPU#=
CPU#=
CPU#=
CPU#=
CPU#=
CPU#=
CPU#=
* GLIC= ONLY FOR GENERAL LICENSE
GLIC=
*
/*
* **************************************************************
* **** LINKING LICENSE MODULE: FLAMLIZ                     ****
* **************************************************************
* **** LINKING PHASE  FLAMLIZ  INTO USER LOADLIBRARY         ****
// LIBDEF PHASE,CATALOG=&FLAMLIB
// LIBDEF OBJ,SEARCH=(&FLAMLIB,IJSYSRS.SYSLIB)
// OPTION ERRS,SXREF,SYM,LIST,NODECK,CATAL
```

```
*  ACTION MAP
   PHASE FLAMLIZ,*
   INCLUDE FLAMLIZ
// EXEC LNKEDT,PARM='AMODE=24,RMODE=24'
/*
```

```
* ***************************************************************
* ****   TEST: FLAM COMPRESSION                              ****
* ***************************************************************
// LIBDEF PHASE,SEARCH=&FLAMLIB
// EXEC FLAM,SIZE=AUTO,PARM='C,FLAMI=&FLAMLIB(FLAMPAR.OBJ),MODE=ADC,FLAC
               MDDN=*DUMMY,END'
/*
/&
* $$ EOJ
```

Das Lizenzprogramm generiert ein neues Lizenz-Objektmember, das dann mit dem LINKAGE-EDITOR zu ausf hrbaren Phasen gebunden wird.

FLAM (VSE)

Benutzerhandbuch

Kapitel 7:

Technische Daten

Inhalt

7. Technische Daten

7.1 Systemumgebung

FLAM V4.1 (VSE) ist ablauff hig ab VSE Rel.3.1.

Komprimate aller Vorg nger-Versionen von FLAM k nnen mit dieser Version dekomprimiert werden. Innerhalb der Versionen 2, 3 und 4 ist FLAM sowohl aufw rts- als auch abw rtskompatibel, dabei wird immer nur der Funktionsumfang der niedrigen Version unterst tzt.

7.2 Speicheranforderungen

Die Komponenten von FLAM ben tigen jeweils statischen Speicher f r den Objektcode. Dazu werden dynamisch zur Laufzeit Speicherbereiche f r Variable und Arbeitsbereiche angefordert. Zus tzlich werden vom Betriebssystem Ein-/Ausgabepuffer f r Dateien angelegt.

	statisch	dynamisch	Matrix
FLAM / FLAMUP mit Folgemodulen	230 KB	20-80 KB	6-5300 KB
Satzschnittstelle mit Folgemodulen	150 KB	10-40 KB	6-5300 KB

Die angegebenen Werte sind Gr enordnungen. Der dynamische Speicher ist abh ngig von der L nge der zu bearbeitenden S tze und der Dateizugriffsmethode.

7.3 Leistungen

Folgende Beispiele aus Testreihen sollen Anhaltspunkte geben, welche Komprimierungseffekte zu erwarten sind:

typische Anwenderdaten (wie FIBU, MATDAT) 70 - 90%

diverse Listen (wie ASSEMBLER-Listings) 65 - 85%

Datentr ger-Austausch-Dateien (DTAUS) 70%

Grunds tzlich ist der Komprimierungseffekt vom Dateiaufbau und den Satzstrukturen, sowie den Daten selbst abh ngig, au erdem vom Komprimierungsmodus und den verwendeten Parametern.

7.4 Statistik

Bei Parameterangabe SHOW=ALL gibt FLAM / FLAMUP statistische Daten zum Ablauf der Komprimierung/Dekomprimierung aus.

FLAM kann Satz- und Byteanzahlen sowie den Kompressionsgrad ermitteln und protokollieren. Dabei werden bei der Komprimierung die Anzahl der eingegebenen S tze und Bytes, die Anzahl der ausgegebenen S tze und Bytes und der Kompressionsgrad als prozentuales Verh ltnis zwischen ein- und ausgegebenen Datenbytes ermittelt. Die Byteanzahl wird aus den Nettol ngen der Datens tze errechnet, d.h. ohne Ber cksichtigung eines vorhandenen Satzl ngenfeldes.

Der Komprimierungseffekt wird immer aus dem Verh ltnis der eingegebenen zu den ausgegebenen Bytes berechnet.

Bei der Verwendung von Benutzerausg ngen kann durch Ver nderung der Satzanzahl oder L nge die Statistik verf lscht werden.

Bei der Dekomprimierung wird die Anzahl der S tze und Bytes der FLAMFILE ermittelt. Au erdem werden die Anzahl und Bytes der dekomprimierten S tze ausgegeben. Die Zahlen der Komprimierung und Dekomprimierung stimmen berein, wenn keine Benutzerausg nge benutzt werden.

FLAM protokolliert die elapsed time des Vorgangs, das hei t in dieser Zeitangabe sind z.B. auch alle R stzeiten zur Bandmontage enthalten.

Beim Komprimieren und Dekomprimieren von Sammeldateien werden f r alle verarbeiteten Teilkomprimate Zwischenstatistiken mit den Satz- und Byteanzahlen der Original- und Komprimatss tze ausgegeben.

Am Ende einer Sammeldatei wird eine Gesamtstatistik mit den Satz- und Byteanzahlen, dem Komprimierungseffekt und den Zeitangaben ausgegeben. Vor dieser Gesamtstatistik wird der Dateiname der Komprimatsdatei wiederholt; gegebenenfalls wird eine Meldung ausgegeben, dass nicht alle Dateien verarbeitet werden konnten.

Beim Dekomprimieren von Sammeldateien werden nur die Satz- und Byteanzahlen der verarbeiteten Komprimatss tze in die Gesamtstatistik aufgenommen, die Werte f r die Originals tze werden nur in die Zwischenstatistiken f r die Einzeldateien aufgenommen. Bei der Verarbeitung einer Dateimenge wird f r jede Datei die Statistik getrennt ausgegeben. Nur die Zeitangaben erscheinen gemeinsam am Ende des Programmlaufs.

FLAM (VSE)

Benutzerhandbuch

Kapitel 8:

Meldungen

Inhalt

8. Meldungen

8.1 Meldungen des Dienstprogramms

Meldungen werden nur durch das Dienstprogramm FLAM oder auf der Unterprogrammschnittstelle FLAMUP ausgegeben. Unterhalb der Satzschnittstelle FLAMREC erfolgt keine Meldungsausgabe.

Mit dem Parameter MSGDISP kann die Art der Meldungsausgabe bestimmt werden:

MSGDISP=TERMINAL

wird nicht unterst tzt.

MSGDISP=MSGFILE

Die Meldungen werden in eine Datei geschrieben. Der DD-NAME ist standardm ig FLAMMSG und kann mit dem Parameter MSGDDN=<name> ge ndert werden.
z.B. Zuordnung der Meldedatei:
// DLBL FLAMMSG,'Meldungsdatei',,VSAM,CAT=catalog

MSGDISP=SYSTEM

Meldungen werden   ber DTFPR auf $$ LST ausgegeben.

8.2 bersicht

FLAM Meldungen zur Komprimierung

FLM0400	FLAM COMPRESSION VERSION ... ACTIVE
FLM0401	REJECTED. INVALID VALUE: ...
FLM0402	REJECTED. SYNTAX ERROR
FLM0403	REJECTED. INVALID KEYWORD
FLM0404	REJECTED. VALUE NOT DECIMAL
FLM0405	REJECTED. OPERAND IS TOO LONG
FLM0406	INPUT RECORDS/BYTES: ...
FLM0407	OUTPUT RECORDS/BYTES: ...
FLM0408	CPU - TIME: ...
FLM0409	RUN - TIME: ...
FLM0410	DATA SET NAME: ...
FLM0411	DATA SET ORGANIZATION NOT SUPPORTED
FLM0413	COMPRESSION ERRORCODE: ...
FLM0414	FLAMFILE SPLIT ACTIVE
FLM0415	USED PARAMETER: ...
FLM0416	COMPRESSION REDUCTION IN PERCENT: ...
FLM0421	INPUT SUPPRESSED
FLM0422	INPUT DATA SET IS EMPTY
FLM0424	ILLEGAL FUNCTION OR INSUFFICIENT MEMORY
FLM0426	MESSAGE NOT FOUND
FLM0428	RECEIVED: ...
FLM0429	NAME GENERATION ERROR: NUMERIC RANGE OVERFLOW
FLM0431	FLAMFILE SPLIT NO. nn MISSING
FLM0432	FLAMFILE SPLIT SEQUENCE ERROR. FOUND NO. nn, NEED NO. mm
FLM0433	FLAMFILE SPLIT NO. nn IS NOT A CONTINUATION

FLM0435	FLAMFILE MAC: nnnnnnnnnnnnnnn MEMBER MAC:
FLM0440	FLAM COMPRESSION NORMAL END
FLM0441	ERROR IN OPERATION: ...
FLM0442	DMS ERRORCODE: ... DD-NAME: ...
FLM0443	FLAM ERRORCODE: ... DD-NAME: ...
FLM0444	COMPRESSION-LIMIT WARNING
FLM0445	. . . . . . *Nachricht des KMEXITs*
FLM0448	COPYRIGHT (C) 1989-2007 BY LIMES DATENTECHNIK ..
FLM0449	FLAM COMPRESSION TERMINATED WITH ERRORS

FLAM Meldungen zur Dekomprimierung

FLM0448	COPYRIGHT (C) 1989-2007 BY LIMES DATENTECHNIK ..
FLM0450	FLAM DECOMPRESSION VERSION ... ACTIVE
FLM0401	REJECTED. INVALID VALUE: ...
FLM0402	REJECTED. SYNTAX ERROR
FLM0403	REJECTED. INVALID KEYWORD
FLM0404	REJECTED. VALUE NOT DECIMAL
FLM0405	REJECTED. OPERAND IS TOO LONG
FLM0428	RECEIVED: ...
FLM0456	INPUT RECORDS/BYTES: ...
FLM0457	OUTPUT RECORDS/BYTES: ...
FLM0458	CPU - TIME: ...
FLM0459	RUN - TIME: ...
FLM0460	DATA SET NAME: ...
FLM0461	DATA SET ORGANIZATION NOT SUPPORTED
FLM0462	WRITTEN RECORDS/BYTES: ...
FLM0463	DECOMPRESSION ERRORCODE: ...

FLM0465	USED PARAMETER: ...
FLM0468	SPLIT RECORDS / BYTES: …
FLM0469	COMPRESSED FILE FLAM-ID: ...
FLM0470	SPLIT ID: …
FLM0471	OUTPUT SUPPRESSED
FLM0472	INPUT DATA SET IS EMPTY
FLM0474	ILLEGAL FUNCTION OR INSUFFICIENT MEMORY
FLM0475	CRYPTOKEY WRONG OR MISSING
FLM0476	NO. SPLITS EXCEEDS MAXIMUM OF nn
FLM0479	FILE-ATTRIBUTE WAS CHANGED
FLM0480	FILE PARAM OLD: ... NEW : ...
FLM0481	RECORD TRUNCATED
FLM0482	OLD ...
FLM0483	ACTUAL FLAMFILE VERSION NOT SUPPORTED: nn
FLM0485	FLAMFILE MAC: *nnnnnnnnnnnnnnnn* MEMBER MAC :
FLM0488	INPUT WAS NOT COMPRESSED BY FLAM
FLM0490	FLAM DECOMPRESSION NORMAL END
FLM0491	ERROR IN OPERATION: ...
FLM0492	DMS ERRORCODE: ... DD-NAME: ...
FLM0493	FLAM ERRORCODE: ... DD-NAME: ...
FLM0499	FLAM DECOMPRESSION TERMINATED WITH ERRORS

8.3 Auflistung

FLAM Meldungen

FLM0400 FLAM COMPRESSION VERSION ... ACTIVE

Bedeutung Das Komprimierungssystem FLAM wurde aktiviert. FLAM bedeutet: Frankenstein-Limes-Access-Method.

FLAM fi ist ein eingetragenes Warenzeichen.
Copyright ' by limes datentechnik gmbh, 1999.

Reaktion Keine.

FLM0401 REJECTED. INVALID VALUE: ...

Bedeutung Der angegebene Parameter hat einen ung ltigen Wert.

Reaktion Parameter nach der FLAM-Beschreibung korrigieren und neu starten.

FLM0402 REJECTED. SYNTAX ERROR

Bedeutung Die Anweisung kann nicht angenommen werden, da sie einen Syntaxfehler enth lt. Die Anweisung wurde mit der Meldung FLM0428 protokolliert.

Reaktion Anweisung mit richtiger Syntax eingeben.

FLM0403 REJECTED. INVALID KEYWORD

Bedeutung Die Anweisung kann nicht angenommen werden, da sie ein undefiniertes Schl sselwort enth lt. Die richtigen Schl sselworte und ihre Abk rzung sind der Schnittstellenbeschreibung zu entnehmen.

Reaktion Das ung ltige Schl sselwort korrigieren und neu starten.

FLM0404 REJECTED. VALUE NOT DECIMAL

Bedeutung Die Anweisung kann nicht angenommen werden, da die Wertzuweisung f r einen Operanden nicht dezimal ist. Die Anweisung wurde mit FLM0428 protokolliert.

Reaktion Die Anweisung mit dezimaler Wertzuweisung wiederholen.

FLM0405 REJECTED. OPERAND IS TOO LONG

Bedeutung Die Anweisung kann nicht angenommen werden, da die Wertzuweisung f r einen Operanden zu lang ist. Die Anweisung wurde mit FLM0428 protokolliert.

Reaktion Die Anweisung mit richtiger Wertzuweisung wiederholen.

FLM0406 INPUT RECORDS / BYTES: ...

Bedeutung Anzahl der mit FLAM komprimierten Datens tze und Bytes.

Reaktion Keine.

FLM0407 OUTPUT RECORDS / BYTES: ...

Bedeutung Anzahl Datens tze und Datenbytes im Komprimat (FLAMFILE).

Reaktion Keine.

FLM0408 CPU - TIME: ...

Bedeutung Von FLAM bei der Komprimierung verbrauchte CPU-Zeit.

Reaktion Keine.

FLM0409 RUN - TIME: ...

Bedeutung Ablaufdauer der Komprimierung mit FLAM (elapsed time). Darin sind z.B. auch R stzeiten f r B nder enthalten.

Reaktion Keine.

FLM0410 DATA SET NAME: ...

Bedeutung Name der mit FLAM zu komprimierenden Datei (FLAMIN), der Komprimatsdatei (FLAMFILE) oder der Parameterdatei (PARFILE).

Reaktion Keine.

FLM0411 DATA SET ORGANIZATION NOT SUPPORTED

Bedeutung Die Eingabedatei kann nicht komprimiert werden, da FLAM diesen Dateityp nicht unterst tzt.

Reaktion Eine Datei zuweisen, die von FLAM unterst tzt wird.

FLM0413 COMPRESSION ERRORCODE: ...

Bedeutung Abbruch der Komprimierung. Bedeutung der Fehlercodes: (siehe auch Kapitel 8.4)

15 = Satzl nge gr   er als 32763 bzw. negativ

16 = Satzl nge gr   er als Matrixgr sse -4

20 = Unzul ssiger Openmode

21 = Unzul ssige Gr   e des Matrixpuffers

22 = Unzul ssiges Kompressionsverfahren

23 = Unzul ssiger Code in FLAMFILE

24 = Unzul ssige MAXRECORDS Angabe

25 = Unzul ssige Satzl nge

26 = Unzul ssiger Zeichencode

40 = Modul oder Tabelle kann nicht geladen werden

41 = Modul kann nicht aufgerufen werden

42 = Modul kann nicht entladen werden

43 - 49 = Fehlerabbruch durch Exit-Routine

98 = Es wurden nicht alle Dateien bearbeitet

Reaktion In der Regel sind f r FLAM falsche Parameter (siehe Kapitel 3)   bergeben worden. Diese sind zu korrigieren.

Die Fehlercodes 15, 16, 25 und 40 - 49 sind selbsterkl rend.

Bei anderen Fehler-Codes erstellen Sie bitte Fehlerunterlagen und wenden sich an Ihren Vertriebspartner.

FLM0414 FLAMFILE SPLIT ACTIVE

Bedeutung Das Teilen oder Zusammenf gen einer gesplitteten FLAMFILE ist aktiviert.

Reaktion Keine.

FLM0415 USED PARAMETER: ...

Bedeutung Protokoll der benutzten Parameter zur Komprimierung.

Reaktion Keine.

FLM0416 COMPRESSION REDUCTION IN PERCENT: ...

Bedeutung Die Input-Datenbytes wurden um ... Prozent reduziert.

Reaktion Keine.

FLM0421 INPUT SUPPRESSED

Bedeutung Eingabedatei wurde nicht bearbeitet.

Reaktion Keine.

FLM0422 INPUT DATA SET IS EMPTY

Bedeutung Die zu komprimierende Datei ist logisch leer.

Reaktion Keine.

FLM0424 ILLEGAL FUNCTION OR INSUFFICIENT MEMORY

Bedeutung Es wurde eine ung ltige Funktion angefordert oder es steht nicht gen gend Speicherplatz zur Verf gung.

Evtl. hat sich auch seit der Lizenzierung Ihr Rechner ge ndert, so dass FLAM-Aufrufe als ung ltig abgewiesen werden (Error in FLMOPD).

Reaktion Lizenz berpr fen. Speicherplatz berpr fen, gegebenenfalls MAXBUFFER verkleinern.

FLM0426 MESSAGE NOT FOUND

Bedeutung Fehler in den FLAM-Modulen.

Reaktion Bitte Fehlerunterlagen erstellen und den Vertriebspartner informieren.

FLM0428 RECEIVED: ...

Bedeutung Protokoll der bergebenen Komprimierungs-Parameter.

Reaktion Keine.

FLM0429 NAME GENERATION ERROR: NUMERIC RANGE OVERFLOW

Bedeutung Beim Teilen oder Zusammenf gen von Fragmenten einer FLAMFILE kann kein weiterer Dateiname (oder DD-Name) gebildet werden. Z.B. m sste nach Dateinummer 9 die Nummer 10 generiert werden, der Dateiname (DD-Name) enth lt aber nur eine einstellige Ziffernfolge (z.B. NAME1 vorgegeben anstatt NAME01)

Reaktion Mehr Ziffernfolgen im Dateinamen (DD-Namen) angeben.

FLM0431 FLAMFILE SPLIT NO. nn MISSING

Bedeutung Beim Dekomprimieren kann das Fragment Nr. nn einer geteilten FLAMFILE nicht gefunden werden. Die Datei ist z.B. nicht katalogisiert, exklusiv im Zugriff oder die zugeh rige DD-Anweisung fehlt in der JCL.

Reaktion berpr fen Sie den Dateinamen, die Datei katalogisieren, den Lauf sp ter erneut starten.

FLM0432 FLAMFILE SPLIT SEQUENCE ERROR. FOUND NO. nn, NEED NO. mm

Bedeutung Es wurde zur Dekomprimierung das Fragment Nr. mm der geteilten FLAMFILE erwartet. Gefunden wurde aber Nr. nn. Das Fragment ist Teil der zuerst gelesenen Datei, liegt aber in falscher Reihenfolge vor.

Reaktion Die Dateien mit der entsprechenden Ziffernfolge im Dateinamen katalogisieren oder bei Zuweisung ber den DD-Namen in der JCL die Reihenfolge korrigieren.

FLM0433 FLAMFILE SPLIT NO. nn IS NOT A CONTINUATION

Bedeutung Beim Dekomprimieren geh rt das aktuelle Fragment Nr. nn der gesplitteten FLAMFILE nicht zur vorhergehenden

Datei. Es ist zwar Teil einer FLAMFILE, aber geh rt nicht zum zuerst gelesenen Teil dazu.

Reaktion Die zugeh rige Datei zuweisen.

Anmerkung Jeder neue Komprimierungslauf erzeugt auch bei identischer Eingabe eine andere FLAMFILE. Damit sind Fragmente gesplitteter FLAMFILEs von verschiedenen Komprimierungen nicht austauschbar !

FLM0435 FLAMFILE MAC: nnnnnnnnnnnnnnn
MEMBER MAC :

Bedeutung: Protokoll des errechneten Hash-MACs der gesamten FLAMFILE, bzw. jeden Members der Sammel FLAMFILE.

Reaktion: Keine.

Anmerkung: Jede mit AES verschl sselte FLAMFILE wird mit einem Hash-MAC abgeschlossen. Zus tzlich ist jedes Member einer Sammel FLAMFILE separat gesichert. Diese MACs dienen dem Integrit tsschutz auf Matrix-, Member- und FLAMFILE Ebene.

FLM0440 FLAM COMPRESSION NORMAL END

Bedeutung Die Komprimierung mit FLAM wurde normal beendet.

Reaktion Keine.

FLM0441 ERROR IN OPERATION: ...

Bedeutung Bei dieser Funktion ist ein Fehler aufgetreten. Der Fehlercode ist in der nachfolgenden Meldung protokolliert.

FLAMSYN	Syntaxanalyse f r Parametereingabe
FLAMREQM	Speicheranforderung
FLAMFREE	Speicherfreigabe
FLAMSCAN	Analyse einer Auswahl- bzw. Umsetzanweisung f r Dateinamen
FLAMUP	Ablaufsteuerung
WCDxxx	Dateinamen in Wildcardsyntax verarbeiten
DYNxxx	Dynamisches Laden von Modulen und Tabellen

TIOxxx Terminal Ein-/Ausgabe

MSGxxx Meldungsausgabe

TIMxxx Zeitmessung

FIOxxx Datei Ein-/Ausgabe

FLMxxx FLAM Satzschnittstelle

Reaktion Keine.

FLM0442 DMS ERRORCODE: ... DD-NAME: ...

Bedeutung Bei der Verarbeitung der Datei mit dem angegebenen DD-Namen ist ein Fehler aufgetreten.

Reaktion Fehlercode analysieren und Datei entsprechend korrigieren.

FLM0443 FLAM ERRORCODE: ... DD-NAME: ...

Bedeutung Bei der Verarbeitung der Datei mit dem angegebenen DD-Namen ist ein Fehler aufgetreten. Bedeutung der Errorcodes (siehe auch Kapitel 8.4):

30 = Eingabe-Datei leer

31 = Eingabe-Datei nicht vorhanden

32 = Ung ltiger Open Mode

33 = Ung ltiger Dateityp

34 = Ung ltiges Satzformat

35 = Ung ltige Satzl nge

36 = Ung ltige Blockl nge

37 = Ung ltige Schl sselposition

38 = Ung ltige Schl ssell nge

39 = Ung ltiger Dateiname

40 =	Modul oder Tabelle kann nicht geladen werden
43 - 49 =	Fehlerabbruch durch Exit
52 =	Zuviele oder unzul ssige doppelte Schl ssel
98 =	Es wurden nicht alle Dateien bearbeitet

Reaktion Fehlercode analysieren und Datei entsprechend korrigieren.

FLM0444 COMPRESSION-LIMIT WARNING

Bedeutung Komprimierungsergebnis ist schlechter als der eingestellte Grenzwert. (CLIMIT, 3.1.1) Der Return Code 4 ist gesetzt.

Reaktion Keine.

FLM0448 COPYRIGHT (C) 1989-1999 BY LIMES DATENTECHNIK GMBH

Bedeutung Copyright Meldung mit Kundenlizenznummer, bzw. Ablaufdatum bei Testinstallation.

Reaktion Keine.

FLM0449 FLAM COMPRESSION TERMINATED WITH ERRORS

Bedeutung Die Komprimierung wurde mit Fehlern beendet. Ein Return Code von 8, 12 oder 16 ist gesetzt.

Reaktion Keine, bzw. je nach vorangegangener Meldung.

FLM0450 FLAM DECOMPRESSION VERSION ... ACTIVE

Bedeutung Das Dekomprimierungssystem FLAM wurde aktiviert. FLAM bedeutet: Frankenstein-Limes-Access-Method. FLAMfi ist ein eingetragenes Warenzeichen. Copyright ' by limes datentechnik gmbh, 1999.

Reaktion Keine.

FLM0456 INPUT RECORDS/BYTES: ...

Bedeutung Anzahl Datens tze und Datenbytes im Komprimat (FLAMFILE).

Reaktion Keine.

FLM0457 OUTPUT RECORDS/BYTES: ...

Bedeutung Anzahl der mit FLAM dekomprimierten Datens tze und Datenbytes.

Reaktion Keine.

FLM0458 CPU - TIME: ...

Bedeutung Von FLAM bei der Dekomprimierung verbrauchte CPU-Zeit.

Reaktion Keine.

FLM0459 RUN - TIME: ...

Bedeutung Ablaufdauer der Dekomprimierung mit FLAM (elapsed time). Darin sind z.B. auch R stzeiten f r B nder enthalten.

Reaktion Keine.

FLM0460 DATA SET: ...

Bedeutung Name der mit FLAM zu dekomprimierenden Datei (FLAMFILE), oder der Ausgabedatei (FLAMOUT).

Reaktion Keine.

FLM0461 DATA SET ORGANIZATION NOT SUPPORTED

Bedeutung Die Ausgabedatei kann nicht erzeugt werden, da FLAM diesen Dateityp nicht unterst tzt.

Reaktion Eine Ausgabedatei zuweisen, die von FLAM unterst tzt wird.

FLM0462 WRITTEN RECORDS/BYTES: ...

Bedeutung Anzahl der geschriebenen Datens tze und Bytes. Differenz zu FLM0457 entsteht bei Dateikonvertierung.

Reaktion Keine.

FLM0463 DECOMPRESSION ERRORCODE: ...

Bedeutung Die Dekomprimierung wurde mit dem Fehlercode ... beendet (Siehe auch Kapitel 8.4).

10 = Datei keine FLAMFILE

11 = FLAMFILE Formatfehler

12 = Satzl ngenfehler

13 = Dateil ngenfehler

14 = Checksummenfehler

20 =	Unzul ssiger Openmode
21 =	Unzul ssige Gr e des Matrixpuffers
22 =	Unzul ssiges Kompressionsverfahren
23 =	Unzul ssiger Code in FLAMFILE
24 =	Unzul ssige MAXRECORDS Angabe
25 =	Unzul ssige Satzl nge
26 =	Unzul ssiger Zeichencode
40 =	Modul oder Tabelle kann nicht geladen werden
41 =	Modul kann nicht aufgerufen werden
42 =	Modul kann nicht entladen werden
43 - 49 =	Fehlerabbruch durch Exit-Routine
52 =	Zuviele oder unzul ssige doppelte Schl ssel
57 =	Unzul ssige Teilkomprimatsl nge
60 - 78	FLAM-Syntaxfehler (siehe 3.3.11 FLMGET)
96 =	Keinen Dateinamen gefunden
98 =	Es wurden nicht alle Dateien bearbeitet

Reaktion

Bei Fehlercode 10 - 14 liegt FLAMFILE nicht mehr im urspr nglichen Zustand vor. Die Fehlercode 40 - 49 sind selbsterkl rend.

Bei Fehlercode 60 - 78 bitte Fehlerunterlagen erstellen und den Vertriebspartner informieren.

FLM0465 USED PARAMETER: ...

Bedeutung Protokoll der benutzten Dekomprimierungsparameter.

Reaktion Keine.

FLM0468 SPLIT RECORDS / BYTES: …

Bedeutung

Zahl der Datens tze und Bytes im aktuellen Fragment der gesplitteten FLAMFILE. Beim Splitt werden jeweils Daten zur Steuerung und Kontrolle eingef gt, so dass die Summe der Datens tze und –bytes gr er ist als das ‚eigentliche' Komprimat (Meldungen FLM0407, FLM0456).

Reaktion

Keine.

FLM0469 COMPRESSED FILE FLAM-ID: ...

Bedeutung

FLAM-Systemcode der Originaldatei.

0080	MS-DOS
0101	IBM MVS
0102	IBM VSE
0103	IBM VM
0104	IBM 81xx
0105	IBM DPPX/370
0106	IBM AIX
02xx	UNISYS
0301	DEC VMS
0302	DEC ULTRIX
0401	SIEMENS BS2000
0402	SIEMENS SINIX
0403	SIEMENS SYSTEM V
0501	NIXDORF 886x
0502	NIXDORF TARGON
06xx	WANG
07xx	PHILLIPS

08xx	OLIVETTI
09xx	TANDEM
0Axx	PRIME
0Bxx	STRATUS
0E02	APLLE A/UX
11xx	INTEL 80286
12xx	INTEL 80386
13xx	INTEL 80486
15xx	Motorola 68000
xx04	UNIX

Reaktion Keine.

FLM0470 SPLIT ID: …

Bedeutung Jedes Fragment einer parallel gesplitteten FLAMFILE erh lt eine eindeutige Kennung, die zur Authentifizierung verwendet werden kann. Der zugeh rige Dateiname wurde mit FLM0410, FLM0460 protokolliert.

Reaktion Keine.

FLM0471 OUTPUT SUPPRESSED

Bedeutung Ausgabedatei nicht verarbeitet.

Reaktion Keine.

FLM0472 INPUT DATA SET IS EMPTY

Bedeutung Bei der zu dekomprimierenden Datei (FLAMFILE) handelt es sich um eine logisch leere Datei.

Reaktion Zur Dekomprimierung eine FLAMFILE zuweisen.

FLM0474 ILLEGAL FUNCTION OR INSUFFICIENT MEMORY

Bedeutung

Es wurde eine ung ltige Funktion angefordert oder es steht nicht gen gend Speicherplatz zur Verf gung. Evtl. hat sich seit der Lizenzierung Ihr Rechner ge ndert, so dass FLAM-Aufrufe als ung ltig abgewiesen werden (Error in FLMOPS). Oder FLAM ist gar nicht lizenziert.

Reaktion

Lizenzierung, Speicherplatz berpr fen.

FLM0475

PASSWORD WRONG OR MISSING

Bedeutung:

Bei der Dekomprimierung wurde das Passwort nicht angegeben oder es wurde bei der Komprimierung ein anderes verwendet.

FLM0479

FILE-ATTRIBUTE CHANGED

Bedeutung

F r die Ausgabedatei gelten andere Dateiattribute als f r die Originaldatei. Es erfolgt eine Konvertierung in die neuen Angaben.

Reaktion

Keine, bzw. Ausgabedatei anders definieren.

FLM0480

FILE PARAM OLD: ... NEW: ...

Bedeutung

Auflistung der Original-Dateiattribute und der bei der Dekomprimierung angegebenen.

Reaktion

Keine, bzw. Ausgabedatei anders definieren.

FLM0481

RECORD TRUNCATED

Bedeutung

Ein Satz wurde verk rzt. Die dekomprimierte Datei enth lt einen (oder mehrere) S tze, die l nger sind als die im Dateikatalog definierte Satzl nge.

Bei TRUNCATE=NO wird das Programm mit Fehler beendet.

Reaktion

F r eine Konvertierung ist der Programmlauf mit dem FLAM-Parameter TRUNCATE=YES zu starten.

Eine Datei mit gr erer Satzl nge zuweisen.

FLM0482

OLD ...

Bedeutung

Protokoll des FLAM-Fileheaders.

OLD DSN : Dateiname der Originaldatei

OLD CODE : Original-Datei-Code

OLD DSORG : Original-Datei-Organisation

OLD RECFORM : Original-Datei-Format

OLD RECSIZE : Original-Datei-Satzl nge

OLD BLKSIZE : Original-Datei-Blockgr sse

OLD KEYPOS : Original-Datei-Schl ssel-Position

OLD KEYLEN : Original-Datei-Schl ssel-L nge

Reaktion Keine.

FLM0488 INPUT WAS NOT COMPRESSED BY FLAM

Bedeutung Die Eingabe wurde nicht mit FLAM komprimiert. RC wird auf 88 gesetzt.

Reaktion Eine mit FLAM komprimierte Datei zuweisen.

FLM0490 FLAM DECOMPRESSION NORMAL END

Bedeutung Die Dekomprimierung mit FLAM wurde normal beendet.

Reaktion Keine.

FLM0491 ERROR IN OPERATION: ...

Bedeutung Bei dieser Funktion ist ein Fehler aufgetreten, der Fehlercode ist in der nachfolgenden Meldung protokolliert.

Reaktion Keine.

FLM0492 DMS ERRORCODE: ... DD-NAME: ...

Bedeutung Bei Verarbeitung der Datei mit dem angegebenen DD-Namen, ist ein Fehler aufgetreten.

Reaktion Fehlercode analysieren und Datei entsprechend korrigieren.

FLM0493 FLAM ERRORCODE: ... DD-NAME: ...

Bedeutung

Bei der Verarbeitung der Datei mit dem angegebenen DD-Namen ist ein FLAM-Fehler aufgetreten. Bedeutung der Errorcodes (siehe auch Kapitel 8.4):

30 = Eingabe-Datei leer

31 = Eingabe-Datei nicht vorhanden

32 = Ung ltiger Open Mode

33 = Ung ltiger Dateityp

34 = Ung ltiges Satzformat

35 = Ung ltige Satzl nge

36 = Ung ltige Blockl nge

37 = Ung ltige Schl sselposition

38 = Ung ltige Schl ssell nge

39 = Ung ltiger Dateiname

Reaktion

Fehlercode analysieren und Datei entsprechend korrigieren.

FLM0499

FLAM DECOMPRESSION TERMINATED WITH ERRORS

Bedeutung

Die Dekomprimierung mit FLAM wurde mit Fehler beendet. Condition Code ist auf 4, 8, 12 oder 16 gesetzt.

Reaktion

Fehler analysieren.

8.4 FLAM-Returncodes

Durch FLAM werden an den verschiedenen Schnittstellen (FLAMUP, FLAMREC, USERIO) bestimmte Ausnahmesituationen und Fehler durch systemneutrale Returncodes gemeldet.

Bei Fehlercodes, die sich auf **Dateioperationen** beziehen, wird die Datei im h chstwertigen Byte des vierstelligen Returncode-Feldes markiert:

X'AF' Fehler bei Zugriff auf FLAMOUT
X'CF' FLAMPAR
X'EF' FLAMIN
X'FF' FLAMFILE

Diese Kennzeichen werden von FLAM/FLAMUP zur passenden Meldungsausgabe verwendet.

Die restlichen drei Bytes entsprechen dem Fehlercode der entsprechenden Datei-Zugriffsmethode (wie z.B. VSAM, POWER).

Fehlercodes bei **Verletzung der Security** werden durch Kennzeichen im 2. Byte eingeleitet: 00kkmmmm.
kk bezeichnet den Fehlerort, mit kk =

1 Header
2 Segment
3 Membertrailer
4 Filetrailer

Mit mmmm wird der Fehler selbst beschrieben (Sedezimal):

0001 MAC1, Mac ber das Komprimat
0002 MAC2, Verkettungs MAC
0004 MAC3, Mac ber Macs
0010 Daten fehlen
0020 Daten eingef gt
0040 Daten aktualisiert (update)
0080 Satzz hler Komprimat
0100 Bytez hler Komprimat
0200 Satzz hler Originaldaten
0400 Bytez hler Originaldaten
0800 Verkettung bei FLAM-Verschl sselung

Security-Fehler k nnen nur bei der Dekomprimierung auftreten. Sie k nnen ggf. mit dem Parameter SECURE-INFO=IGNORE den Fehler ignorieren, Folgefehler sind aber nicht auszuschlie en. Bei Positionieren in die FLAMFILE mit anschlie ender Dekomprimierung eines Members muss SECUREINFO=MEMBER angegeben

werden (ansonsten Fehlercode X'00030002, d.h. Fehler der Memberverkettung).

Die nachfolgenden Werte sind Dezimalzahlen

Returncode

0 Die Funktion ist vollst ndig ausgef hrt.

-1 Die Funktion ist nicht ausgef hrt, weil sie im Zusammenhang nicht zul ssig ist (z.B. FLMGET ohne erfolgreiches FLMOPN) oder weil beim ffnen einer Datei nicht ausreichend Speicher zur Verf gung steht.

Returncodes zwischen 1 und 9 sind Warnungen.

Die Funktion ist teilweise ausgef hrt. Der Benutzer muss entscheiden, ob das Ergebnis richtig oder falsch ist.

1 Ein Satz wird auf die L nge des Satzpuffers verk rzt; die Daten k nnen in der angegebenen L nge verarbeitet werden.

2 Beim Lesen wird das Dateiende erreicht; es werden keine Daten bergeben.

3 In einer relativen Datei wird eine L cke gefunden; die Satzl nge ist Null.

4 Beim Konvertieren eines Satzes in fixes Format wird der Satz mit F llzeichen aufgef llt.

5 In einer indexsequentiellen Datei ist beim Lesen ein Schl ssel nicht vorhanden bzw. beim Schreiben ung ltig. Die sequentielle Leseposition steht auf dem Satz mit den n chsth heren Schl ssel.

Beim Positionieren ist die angegebene Position nicht vorhanden bzw. die gew nschte Positionierung ist nicht m glich. Die aktuelle Position wird nicht ver ndert.

Beim L schen ist kein aktueller Satz vorhanden.

6 In einer Sammeldatei beginnt beim Lesen eine neue Datei; es werden keine Daten bergeben. Gegegebenenfalls kann der Fileheader gelesen werden. Die sequentielle Leseposition steht auf dem ersten Satz der neuen Datei.

7 Passwort nicht angegeben. Die FLAMFILE wurde mit einem Passwort komprimiert.
Das Passwort kann mit FLMPWD bergeben werden.

8 unbenutzt

9 FLAMUP bzw. FLAM meldet beim Komprimieren mit eingeschalteter Statistik, dass das Komprimat gr er als das Original ist (Expansion).

Returncodes ber 10 sind Fehler.

Die Funktion ist nicht ausgef hrt bzw. wurde abgebrochen. Eine Ausnahme bildet der Returncode 98 bei FLAMUP bzw. FLAM.

10 Beim Dekomprimieren wird die Eingabedatei nicht als FLAM-Komprimat erkannt. Bereits der Anfang der Datei ist derart verf lscht, dass die FLAM-Syntax nicht mehr erkennbar ist.

M gliche Ursachen f r diesen Fehler sind:

Die Eingabedatei ist kein Komprimat bzw. wurde nicht mit FLAM komprimiert.

Bereits der erste Satz ist verk rzt bzw. vor dem Anfang des FLAM-Komprimats sind Daten eingef gt.

H ufig wird dieser Fehler durch falsch eingestellte File Transfers verursacht:

Beim bertragen von 8-Bit-Komprimaten wird ein File-Transfer f r abdruckbare Daten benutzt und damit die Zeichen des Komprimats verf lscht.

Beim bertragen von indexsequentiellen Komprimatsdateien von DEC-VMS auf andere Systeme wie MVS, BS2000 usw. muss die Schl ssell nge der Komprimatsdatei um die Satz- und Blockz hler (1, 2 bzw. 4 Bytes) vergr ert werden.

Beim bertragen werden Komprimatss tze verk rzt, verl ngert bzw. umgebrochen.

Hinweis: Ein Teil dieser Transformationen wird von FLAM ab der Version 2.7 erkannt und automatisch kompensiert.

Das Auff llen mit gleichen Zeichen wird f r alle Kompressionsverfahren toleriert.

Bei 8-Bit-Komprimaten ist ein Umbruch der Komprimatss tze m glich, sofern bei der Dekomprimierung kein Exit f r die Komprimatss tze (EXD20) aktiv ist.

11 Das Format der FLAMFILE ist fehlerhaft.

Beim Dekomprimieren einer FLAMFILE sind Fehler in der Komprimatssyntax erkannt worden. Beispielsweise k nnen vollst ndige Komprimatss tze fehlen bzw. Header sind verf lscht.

12 Ein Komprimatssatz ist verk rzt, so dass ein Teil der Komprimatsdaten fehlt.

13 Die Komprimatsdatei ist verk rzt. Es fehlen vollst ndige Komprimatss tze am Dateiende. Dieser Fehler kann beim Erzeugen, Kopieren bzw. bertragen von Komprimatsdateien entstehen, wenn nicht ausreichend Speicherplatz f r die Komprimatsdatei zur Verf gung steht und dadurch die Komprimierung, das Kopieren bzw. der File Transfer vorzeitig beendet wird. Jeder andere Abbruch dieser Verarbeitungen kann ebenfalls eine unvollst ndige Komprimatsdatei hinterlassen.

14 Die Checksumme eines Komprimatssatzes ist falsch. Die Komprimatsdatei ist durch Umcodierung oder einen anderen Eingriff verf lscht.

15 FLAM kann nur S tze bis zu einer maximalen Satzl nge von 32.764 Bytes verarbeiten. Die Originaldatei enth lt mindestens einen l ngeren Satz und kann deshalb nicht komprimiert werden.

16 Die Matrixgr e muss um mindestens 4 Byte gr er sein als die gr te Satzl nge in der Originaldatei. F r gute Kompressionseffekte sollte die Matrixgr e mindestens 16mal die Satzl nge sein. Die Datei kann mit gr erem Matrixpuffer erneut komprimiert werden.

17 unbenutzt

18 unbenutzt

19 unbenutzt

20 Unzul ssiger OPENMODE.

Nur indexsequentielle Komprimatsdateien k nnen mit dem OPENMODE = INOUT ge ffnet werden. Sequentielle Komprimatsdateien k nnen nur gelesen (INPUT) bzw. geschrieben werden (OUTPUT).

21 Unzul ssige Gr e des Matrixpuffers.

Beim Dekomprimieren kann der notwendige Matrixpuffer wegen Speicherplatzmangel nicht angefordert werden. Wenn nicht mehr Speicherplatz zur Verf gung gestellt werden kann, muss die Originaldatei mit einem kleineren Matrixpuffer komprimiert werden.

Hinweis: Ab der Version 2.5 wird die doppelte Gr e des Matrixpuffers ben tigt. Gegebenenfalls kann die Komprimatsdatei mit der Version 2.1 dekomprimiert werden, um sie danach mit einem kleineren Matrixpuffer erneut zu komprimieren.

22 Unzul ssiges Kompressionsverfahren.

Das Komprimat ist mit einer neueren FLAM-Version mit einem von dieser Version noch nicht unterst tzten Kompressionsverfahren erzeugt worden.

23 Unzul ssiger Code in FLAMFILE.

Das Komprimat ist in einem Zeichencode (weder ASCII noch EBCDIC) erstellt worden, der von dieser FLAM-Version noch nicht unterst tzt wird.

24 Unzul ssige maximale Satzanzahl.

Der Parameter MAXRECORDS bzw. MAXREC ist ung ltig. Erlaubt sind bei MODE=CX7, CX8, VR8 1 bis 255, bei MODE=ADC 1 bis 4095.

25 Unzul ssige Satzl nge.

Der Parameter MAXSIZE enth lt einen Wert kleiner als 80 bzw. gr er als 32.768 f r 8-Bit-Komprimate. Bei CX7 darf MAXSIZE nicht gr er als 4096 sein.

26 Unzul ssiger Zeichencode.

Die Originaldaten haben einen Zeichencode (weder ASCII noch EBCDIC), der von dieser FLAM-Version noch nicht unterst tzt wird.

27 unbenutzt

28 unbenutzt

29 Es wurde kein oder ein falsches Passwort bergeben.

30 Eingabedatei ist leer. Die Eingabedatei ist vorhanden, aber ohne Inhalt.

31 Eingabedatei ist nicht vorhanden.

32 Ung ltiger OPEN-Mode.

Die Datei kann mit dem gew nschten OPEN-Mode nicht ge ffnet werden. Z.B. kann eine sequentielle Datei nicht zum ˜ ndern ge ffnet werden.

33 Ung ltiger Dateityp.

Das gew nschte Dateiformat kann von FLAM nicht bzw. noch nicht verarbeitet werden.

34 Ung ltiges Satzformat.

Das Satzformat kann von FLAM nicht verarbeitet werden oder es ist f r das angegebene Dateiformat nicht zugelassen.

35 Ung ltige Satzl nge.

Die Satzl nge kann von FLAM nicht verarbeitet werden oder sie ist f r das angegebene Dateiformat und Satzformat nicht zugelassen.

36 Ung ltige Blockl nge.

Die Blockl nge kann von FLAM nicht verarbeitet werden oder sie ist f r das angegebene Dateiformat und Satzformat nicht zugelassen.

37 Ung ltige Schl sselposition.

Bei einer indexsequentiellen FLAMFILE ist die Schl sselposition ungleich 1. F r eine Originaldatei ist die Schl sselposition f r das angegebene Dateiformat nicht zugelassen.

38 Ung ltige Schl ssell nge.

Die Schl ssell nge kann von FLAM nicht verarbeitet werden oder ist f r das angegebene Dateiformat und Satzformat nicht zugelassen.

39 Ung ltiger Dateiname.

Der Dateiname ist in keiner g ltigen Schreibweise f r eine Datei oder ein Bibliothekselement angegeben bzw. es ist eine Wildcard-Angabe f r eine Menge von Dateien und Bibliothekselementen unzul ssig bzw. von FLAM nicht verarbeitbar.

40 Modul oder Tabelle kann nicht geladen werden.

Ein Benutzerausgang bzw. eine bersetzungstabelle kann nicht geladen werden. M glicherweise ist die Bibliothek nicht zugewiesen.

41 Modul kann nicht aufgerufen werden.

Ein Benutzerausgang kann nicht aufgerufen werden.

42 Modul oder Tabelle kann nicht geladen werden.

43 - 49 Fehlerabbruch durch Exit-Routine.

Ein Benutzerausgang hat den Returncode 16 bzw. einen unzul ssigen Returncode zur ckgegeben.

51 unbenutzt

52 Zuviele oder unzul ssige doppelte Schl ssel.

Beim Komprimieren in eine indexsequentielle FLAMFILE enth lt das Original doppelte Schl ssel, obwohl beim ffnen der FLAMFILE in dem Feld KEYFLAGS der Schl sselbeschreibung KEYDESC keine doppelten Schl ssel zugelassen sind. Oder die Anzahl doppelter Schl ssel im Original ist gr er als 255 * MAXSIZE.

53 unbenutzt

56 unbenutzt

57 Unzul ssige Teilkomprimatsl nge.

Das Komprimat einer Matrix ist in mehreren Teilen mit eigenen L ngenfeldern abgelegt. Beim Dekomprimieren wird eine Inkonsistenz dieser L ngenfelder erkannt, ohne dass eine ung ltige Checksumme gefunden wurde.

Dieser Fehler tritt auf, wenn in einer Komprimatsdatei vollst ndige S tze gel scht wurden.

58 unbenutzt

59 unbenutzt

60 - 78 Die Fehler 60 bis 78 beschreiben alle Fehler im Komprimat.

Diese Fehler dienen zur Erkennung von Programmfehlern in FLAM selbst und d rfen deshalb im Betrieb nicht auftreten.

Da mit Hilfe von Checksummen nur mit einer bestimmten Wahrscheinlichkeit eine Verf lschung in einer Komprimatsdatei erkannt wird, kann in seltenen F llen unzutreffenderweise ein Dekompressionsfehler gemeldet werden, obwohl eine Verf lschung vorliegt.

Das Auftreten eines Dekompressionsfehlers sollte unter Beif gung von Fehlerunterlagen an den Hersteller gemeldet werden.

79 unbenutzt

80 Syntaxfehler bei Parametereingabe.

Der Parameterstring ist syntaktisch falsch. Wenn mehrere Parameter auf einmal bergeben wurden, kann durch die Verk rzung des Parameterstrings um jeweils einen Parameter der Fehler eingegrenzt werden.

81 Unbekanntes Schl sselwort.

Im Parameterstring ist ein unbekanntes Schl sselwort enthalten bzw. durch einen Syntaxfehler wird ein Parameterwert als Schl sselwort interpretiert.

82 Unbekannter Parameterwert.

Bei einem Parameter mit einem festen Wertevorrat wie MODE ist ein unzul ssiger Wert angegeben worden.

83 Parameterwert nicht dezimal.

Bei einem Parameter der Zahlen als Wertevorrat hat, ist keine Zahl angegeben worden.

84 Parameterwert zu lang.

Bei einem Parameter ist die Wertangabe zu lang. Zahlenwerte d rfen maximal 8 Zeichen lang sein. Ebenso d rfen feste Werte maixmal 8 Zeichen lang sein. Bei Parametern, die Namen enthalten d rfen, sind die L ngen in der Parameterbeschreibung angegeben. Linknamen, Modulnamen und Namen von Tabellen d rfen maximal 8 Zeichen lang sein. Dateinamen f r einzelne Dateien und als Wildcard-Angaben d rfen maximal 54 Zeichen lang sein.

85 - 95 unbenutzt

96 Keinen Dateinamen gefunden, bzw. Fehler beim Ermitteln von Dateinamen. Dieser Fehler kann bei der Komprimierung im Zusammenhang mit Dateinamensangaben in Wildcard-Syntax oder bei Dateilisten auftreten.

Bei der Dekomprimierung wurde eine Auswahl- oder Umsetzvorschrift f r die Ausgabe vorgegeben und die FLAMFILE enth lt keinen Namen der Originaldatei (durch HEADER=NO oder FILEINFO=NO bei der Komprimierung).

97 unbenutzt

98 Nicht alle Dateien wurden bearbeitet.

Bei der Verarbeitung von Sammeldateien wurden nicht alle Dateien bearbeitet, weil beim ffnen der Originaldateien Fehler erkannt wurden. Alle Dateien, die bearbeitet wurden, sind fehlerfrei bearbeitet.

99 - 998 unbenutzt

999 siehe -1

8.5 Return Codes

Zur Ablaufsteuerung werden bei der Komprimierung durch FLAM folgende Return Codes gesetzt:

$RC=0 Die Komprimierung war fehlerfrei

$RC(4) CLIMIT berschritten.

$RC=6 Es konnten nicht alle Dateien komprimiert werden.

$RC=8 Fehler einfacher Art (wie falsche Parameter) wurden erkannt.

Hinweis: Anhand der Zusatzmeldungen kann der Anwender den Fehler bereinigen.

$RC=12 In der Regel liegen DMS Zugriffsfehler vor.

Hinweis: Anhand der Zusatzmeldungen kann der Anwender den Fehler bereinigen.

$RC=16 Fehler beim Zugriff auf die FLAMFILE oder Komprimierungsfehler.

Hinweis: Kann der Anwender den Fehler nicht beheben, so verst ndigen Sie bitte Ihren Vertriebspartner.

Zur Ablaufsteuerung werden bei der Dekomprimierung durch FLAM folgende Return Codes gesetzt:

$RC=0 Die Dekomprimierung war fehlerfrei

$RC=6 Es konnten nicht alle Dateien dekomprimiert werden.

$RC=8 Fehler einfacher Art (wie falsche Parameter) wurden erkannt.

Hinweis: Anhand der Zusatzmeldungen kann der Anwender den Fehler bereinigen.

$RC=12 In der Regel liegen DMS Zugriffsfehler vor.

Hinweis: Anhand der Zusatzmeldungen kann der Anwender den Fehler bereinigen.

$RC=16 Fehler beim Zugriff auf die FLAMFILE oder Dekomprimierungsfehler

Hinweis: Kann der Anwender den Fehler nicht beheben, so verst ndigen Sie bitte Ihren Vertriebspartner.

$RC=88 Die zugewiesene Datei ist keine FLAMFILE.

Hinweis: Kann zur Identifikation von nicht komprimierten Data Sets in Batch-Jobs genutzt werden.

Nur bei Returncode 0 ist eine Dekomprimierung ordnungsgem   erfolgt. Genauere Hinweise zu Fehlern sind den Fehlermeldungen FLM04xx in Kapitel 8.3 zu entnehmen.

8.6 DMS-Errorcodes

Data Management System ERRORCODES:

CODEFORMAT 4-BYTES H' **zz xx yy cc** '

```
zz = File-ID
          EF = ORIGINALDATEI EINGABE (FLAMIN)
          AF = ORIGINALDATEI AUSGABE (FLAMOUT)
          FF = KOMPRIMATSDATEI FLAMFILE
          CF = PARAMETERDATEI
          DF = MESSAGEDATEI


xx = Subroutine-ID
          00 = VSAM OPEN/GET/PUT/POINT/ERASE (ESDS/KSDS/RRDS)
          01 = LOAD DTF-TABLE           DTFSD/DTFMT/SAM-ESDS
          02 = OPEN VTOC (CVH)          DTFSD
          03 = PROCESS VTOC (CVH)       DTFSD
          04 = CLOSE VTOC (CVH)         DTFSD
          05 = GET RECORD               DTFSD/DTFMT/SAM-ESDS
          06 = PUT RECORD               DTFSD/DTFMT/SAM-ESDS
          07 = OPEN FILE                DTFSD/DTFMT/SAM-ESDS
          08 = POWER ERROR CODES
          09 = LIBR ERROR CODES
          0A = WILDCARD ALLGEM.ERROR VON UNTERROUTINEN
          8X = WILDCARD ERROR FLAG
                   B'1XXX XXXX'  X: USED FROM DMS INTERFACE
                   SAMPLE: B'1000 1000'  = POWER ERROR IN
                                           WILDCARD.
                   OR      B'1000 0001'  = LOAD ERROR IN
                                           WILDCARD.
          0B = FLAM LABEL ERROR IN PROGRAMM FLMLABEL
                 UNTERFUNKTIONEN YY:
                  01 = S$GETLBL
                  02 = S$GETNXT
                  03 = S$SRCLBL
                  04 = S$LOCGRP
                  05 = S$MODGRP
                  06 = S$GETNXL
                  07 = S$REPLBL
                  08 = S$ADDLBL
                  09 = S$ADDNXL
                  0A = S$ENDLBL
                  0B = S$DELLBL
                  0F = S$CLRLBL
                  10 = S$VOLINF
          0C = ERROR BEI CATALOG INFO UEBER FLCATINF
          0D = ERROR ALLOCATE DATA-SET FLCATINF(ALLOCVSM)


yy = Fehler-ID
          REG.15  = NACH VSAM-ERROR ODER 01 BEI DTFSD
          01 = Bei DTFSD/DTFMT/SAM-ESDS ERROR
```

cc = Errorcode

```
Sample Codeliste:
00010104   04  LOAD-ERROR  PHASE NOT FOUND
00010108    08              I/O ERROR DURING LOAD
0001010C    12                INVALID LIB OR SUBLIB
00010110    16                ENTRY OUTSIDE THE PARTITION
00010114    20                SECURITY ??
00010118    24                INCONSISTENT DIRECTORY STATE
0001011C    28                PARTITION IS TOO SMALL

00020108    08  OVTOC-ERROR VOLUME NOT MOUNTED OR INVALID

00030104    04  PVTOC-ERROR I/O ERROR READING VOL1
0003010C    12                I/O ERROR READING VTOC
0003012C    44                FORMAT1 NOT FOUND
00030130    48                INVALID READ OR WRITE ADDRESS
00030158    88                FORMAT4 NOT FOUND
0003015C    92                VOL1 LABEL NOT FOUND

000400??          CVTOC-ERROR  ??

00050100          GET-(LOCATE)-ERROR DTFXX

00060100          PUT-ERROR          DTFXX

0007????          ???? = R15 NACH DTFXX-OPENERROR

000004cc        VSAM-ERROR R15=4   cc = ERRORCODE
000008cc        VSAM-ERROR R15=8   cc = ERRORCODE
0008fnxx        POWER ERROR f=FUNCT. n=R15,
                  xx=POWER-RETCODE
0009nnxx        LIBR  ERROR nn=R15, xx=LIBR-RETC
000Annxx        WILDCARD ALLGEM.ERROR VON UNTERROUTINEN
                           nn=ERR.CODE,           xx=REASON
             CODE
                          01=BUFFER TOO SMALL
```

DMS-POWER-ERRORCODES:

```
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
--------- --------- ---- ---- ---------
    |         |      |    |       |___8-BIT PWR-ERRORFLAG BYTE
    |         |      |    |           (IJBXRETC/IJBXREAS/PXPFBKCD)
    |         |      |    |
    |         |      |    |---------- 4-BIT R15 RETURN REGISTER
    |         |      |
    |         |      |                4 = POWER FAILED/TIME ERROR
    |         |      |                8 = POWER ERROR/INV.REQUEST
    |         |      |                1 = SPOOL-CB. IN DATA AREA
    |         |      |                2 = PWR-MESSAGE IN DATAAREA
    |         |      |                (or PXPRETCD)
    |         |      |
    |         |      |--------------- 4-BIT FUNCTION CODE
    |         |
    |         |                       0001 = FUNCT(IDENTIFY)
    |         |                       0010 = FUNCT(CONNECT)
    |         |                       0011 = FUNCT(SENDR)
    |         |                       0100 = FUNCT(DISCONNECT)
    |         |                       1000 = FUNCT(TERNINATION)
    |         |
    |         |
    |         |---------------------- 8-BIT SUBROUTINE-ID
    |                                 POWER ZUGRIFF = X'08'
    |
    |
    |-------------------------------- 8-BIT File-ID (siehe `zz' )
```

FLAM (VSE)

Benutzerhandbuch

Anhang

Anhang

A.1 bersetzungstabellen

	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	.A	.B	.C	.D	.E	.F	
0.	NUL 00	SOH 01	STX 02	ETX 03	1A	HT 09	1A	DEL 7F	1A	1A	1A	VT 0B	FF 0C	CR 0D	SO 0E	SI 0F	**0.**
1.	DLE 10	DC1 11	DC2 12	DC3 13	1A	1A	BS 08	1A	CAN 18	EM 19	1A	1A	FS 1C	GS 1D	RS 1E	US 1F	**1.**
2.	1A	1A	1A	1A	1A	LF 0A	ETB 17	ESC 1B	1A	1A	1A	1A	1A	ENQ 05	ACK 06	BEL 07	**2.**
3.	1A	1A	SYN 16	1A	1A	1A	1A	EOT 04	1A	1A	1A	1A	DC4 14	NAK 15	1A	SUB 1A	**3.**
4.	SP 20	1A	1A	1A	1A	1A	1A	1A	1A	1A	[5B	. 2E	< 3C	(28	+ 2B	! 21	**4.**
5.	& 26	1A	1A	1A	1A	1A	1A	1A	1A	1A	] 5D	$ 24	* 2A	) 29	; 3B	5E	**5.**
6.	- 2D	/ 2F	1A	1A	1A	1A	1A	1A	1A	1A	ƒ 7C	, 2C	% 25	_ 5F	> 3E	? 3F	**6.**
7.	1A	1A	1A	1A	1A	1A	1A	1A	1A	60	: 3A	# 23	@ 40	’ 27	= 3D	" 22	**7.**
8.	1A	a 61	b 62	c 63	d 64	e 65	f 66	g 67	h 68	i 69	1A	1A	1A	1A	1A	1A	**8.**
9.	1A	j 6A	k 6B	l 6C	m 6D	n 6E	o 6F	p 70	q 71	r 72	1A	1A	1A	1A	1A	1A	**9.**
A.	1A	~ 7E	s 73	t 74	u 75	v 76	w 77	x 78	y 79	z 7A	1A	1A	1A	1A	1A	1A	**A.**
B.	1A	1A	1A	1A	1A	1A	1A	1A	1A	1A	1A	1A	1A	1A	1A	1A	**B.**
C.	{ 7B	A 41	B 42	C 43	D 44	E 45	F 46	G 47	H 48	I 49	1A	1A	1A	1A	1A	1A	**C.**
D.	} 7D	J 4A	K 4B	L 4C	M 4D	N 4E	O 4F	P 50	Q 51	R 52	1A	1A	1A	1A	1A	1A	**D.**
E.	\ 5C	1A	S 53	T 54	U 55	V 56	W 57	X 58	Y 59	Z 5A	1A	1A	1A	1A	1A	1A	**E.**
F.	0 30	1 31	2 32	3 33	4 34	5 35	6 36	7 37	8 38	9 39	1A	1A	1A	1A	1A	1A	**F.**
	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	.A	.B	.C	.D	.E	.F	

bersetzungstabelle von EBCDIC nach ASCII

(TRANSLATE = E/A)

	.0	.1	.2	.3	.4	.5	.6	.7	.8	.9	.A	.B	.C	.D	.E	.F	
0.	NUL 00	SOH 01	STX 02	ETX 03	EOT 37	ENQ 2D	ACK 2E	BEL 2F	BS 16	HT 05	LF 25	VT 0B	FF 0C	CR 0D	SO 0E	SI 0F	**0.**
1.	DLE 10	DC1 11	DC2 12	DC3 13	DC4 3C	NAK 3D	SYN 32	ETB 26	CAN 18	EM 19	SUB 3F	ESC 27	FS 1C	GS 1D	RS 1E	US 1F	**1.**
2.	SP 40	! 4F	" 7F	# 7B	$ 5B	% 6C	& 50	’ 7D	(4D	) 5D	* 5C	+ 4E	, 6B	- 60	. 4B	/ 61	**2.**
3.	0 F0	1 F1	2 F2	3 F3	4 F4	5 F5	6 F6	7 F7	8 F8	9 F9	: 7A	; 5E	< 4C	= 7E	> 6E	? 6F	**3.**
4.	@ 7C	A C1	B C2	C C3	D C4	E C5	F C6	G C7	H C8	I C9	J D1	K D2	L D3	M D4	N D5	O D6	**4.**
5.	P D7	Q D8	R D9	S E2	T E3	U E4	V E5	W E6	X E7	Y E8	Z E9	Ž 4A	\ E0	! 5A	^ 5F	_ 6D	**5.**
6.	‘ 79	a 81	b 82	c 83	d 84	e 85	f 86	g 87	h 88	i 89	j 91	k 92	l 93	m 94	n 95	o 96	**6.**
7.	p 97	q 98	r 99	s A2	t A3	u A4	v A5	w A6	x A7	y A8	z A9	{ C0	ƒ 6A	} D0	~ A1	DEL 07	**7.**
8.	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	**8.**
9.	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	**9.**
A.	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	**A.**
B.	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	**B.**
C.	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	**C.**
D.	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	**D.**
E.	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	**E.**
F.	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	3F	**F.**
	.0	**.1**	**.2**	**.3**	**.4**	**.5**	**.6**	**.7**	**.8**	**.9**	**.A**	**.B**	**.C**	**.D**	**.E**	**.F**	

bersetzungstabelle von ASCII nach EBCDIC

(TRANSLATE = A/E)

Erl uterung der Abk rzungen

ACK	=	acknowledge, positive Quittung
BEL	=	bell, Klingel
BS	=	backspace, Korrekturtaste
CAN	=	cancel, ung ltig, Zeilenl scher
CR	=	carriage return, Wagenr cklauf
DC1	=	device control 1, Ausgabe fortsetzen
DC2	=	device control 2
DC3	=	device control 3, Ausgabe anhalten
DC4	=	device control 4
DEL	=	delete, L schzeichen
DLE	=	data link escape, Austritt aus der Datenverbindung
EM	=	end of medium, Datentr gerende
ENQ	=	enquiry, Stationsaufruf
EOT	=	end of transmission, bertragungsende
ESC	=	escape, R cksprung
ETB	=	end of transmission block, Datenblockende
ETX	=	end of text, Textende
FF	=	form feed, Formularvorschub
FS	=	file separator, Dateitrennung
GS	=	group separater, Gruppentrennung
HT	=	horizontal tabulation, Tabulatorzeichen
LF	=	line feed, Zeilenvorschub
NAK	=	negative acknowledge, negative Quittung
NUL	=	null, keine Operation
RS	=	record separator, Gruppentrennung
SI	=	shift in, zur ckschalten Zeichensatz
SO	=	shift out, umschalten Zeichensatz
SOH	=	start of heading, Vorspannanfang
SP	=	space, Leerzeichen
STX	=	start of text, Textanfang
SUB	=	substitute character, Zeichen ersetzen
SYN	=	synchronous idle, Synchronisierung
US	=	unit separater, Einheitentrennung
VT	=	vertical tabulation